VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Tvorba internetového systému pro prezentaci zájmových bodů - část webového rozhraní

Internet system for Presentation of Points of Interest

2019 Lukáš Klobasa

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Lukáš Klobasa**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Tvorba internetového systému pro prezentaci zájmových bodů - část webového rozhraní
Internet System for Presentation of Points of Interest

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit platformně nezávislou serverovou aplikaci a webové rozhraní pro prezentaci zájmových bodů a navigaci k nim v rámci obcí, měst či rozsáhlejších podniků.

1. Proveďte rešerši možných řešení serverové části včetně databáze a komunikačního protokolu s databází.
2. Realizujte a otestujte serverovou část.
3. Navrhněte webové rozhraní pro komunikaci s databází.
4. Realizujte a otestujte webové rozhraní (v součinnosti s vedoucím práce).

Seznam doporučené odborné literatury:

[1] POKORNÝ, Jaroslav a Michal VALENTA. *Databázové systémy: vysokoškolská učebnice.* Praha: České vysoké učení technické v Praze, 2013. ISBN 978-80-01-05212-9.

[2] LACKO, Ľuboslav. *Vývoj aplikací pro Android.* Přeložil Martin HERODEK. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

[3] HELLMAN, Erik. *Android programming: pushing the limits.* Chichester: Wiley, 2014. ISBN 978-1-118-71737-0.

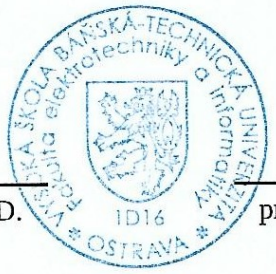
[4] CASTLEDINE, Earle, Myles EFTOS a Max WHEELER. *Vytváříme mobilní web a aplikace pro chytré telefony a tablety.* Přeložil Jakub MUŽÍK. Brno: Computer Press, 2013. ISBN 978-80-251-3763-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019



prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2019 Klobasa .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 1. dubna 2019 Klobasa .....

Rád bych na tomto místě poděkoval mému vedoucímu práce Ing. Janu Skapovi, Ph.D. za konzultace, nápady a přátelský přístup, při tvorbě této bakalářské práce. Dále bych rád poděkoval Matějovi Nedojedlému za zpracování druhé části práce. Nakonec bych rád poděkoval své rodině a přítelkyni za podporu během celého studia a mému dědovi za korekturu gramatiky textu.

Abstrakt

Cílem této práce je vytvoření univerzální platformně nezávislé aplikace pro prezentaci zájmových bodů v rámci okolí a bodů společností či menších podniků, která bude mít své využití v praxi. Aplikace využívá Google Maps API pro zobrazení mapy a práci s ní. Aplikace dokáže vyhledat body, či trasy společnosti na mapě po uživatelské interakci, umožňuje využít navigaci k jednotlivým bodům a zobrazit informace o bodech. Dále uživateli umožňuje vyhledat body zajímavosti v rámci své polohy. Správci společnosti umožňuje správu jednotlivých tras, bodů a svých uživatelů. Práce popisuje jednotlivé technologie, které byly použity pro vytvoření aplikace a pro práci s nimi. Dále popisuje jednotlivé funkcionality aplikace a její testování. Výsledkem práce je funkční responzivní webová aplikace, kterou by společnosti mohly využít pro své zaměstnance k lepší orientaci v rámci svého areálu a okolí.

Klíčová slova: Google Maps API, webová aplikace, zájmové body, navigace, mapa, responzivní

Abstract

The aim of this bachelor thesis is to create a universal platform independent application for presentation of points of interests around companies or smaller enterprises and around user location within cities, which will have its use in practice. The application uses Google Maps API to view and work with the map. Application is able to search for points or routes of the company on the map after user interaction. Application can also use point-to-point navigation and view point information. It also allows the company administrator to manage individual routes, points and their users. The work describes the various technologies that were used to create this application and work with it. It describes the application's functionalities and its testing. The result of this work is a functional responsive web application that companies could use for their employees to better orient themselves within their company and surroundings.

Key Words: Google Maps API, web application, points of interests, navigation, map, responsive

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
PHP	–	Hypertext Preprocessor
XML	–	Extensible Markup Language
HTML	–	Hyper Text Markup Language
JS	–	JavaScript
SQL	–	Structured Query Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
HTML5	–	Hyper Text Markup Language, version 5
CSS	–	Cascade style sheets
GPS	–	Global Positioning System
W3C	–	World Wide Web Consoritum
SMTP	–	Simple Mail Transfer Protocol
FTP	–	File Transfer Protocol
ODBC	–	Open Database Connectivity
GUI	–	Graphical User Interface
CRUD	–	Create, Read, Update, Delete
SDK	–	Software development kit
SSD	–	Solid-state drive
AJAX	–	Asynchronous JavaScript And XML
SSL	–	Secure Sockets Layer

Seznam obrázků

Úvod

Cílem této práce je vytvoření platformně nezávislé webové aplikace, která by se dala v praxi využít v rámci společností či menších podniků. Práce je většího rozsahu, kdy je rozdělena na 2 části, konkrétně část webového rozhraní a Android aplikaci. V této práci je podrobně popsána první část práce. Tato aplikace by měla být uživatelsky přívětivá a responzivní. Snaha byla kladena na to, aby uživatel mohl aplikaci jednoduše používat a nepřipadala mu příliš složitá na ovládání. Dále byl kladen důraz na to, aby webová aplikace byla svým vzhledem shodná v jakémkoli prohlížeči, ať už se jedná o mobilní zařízení či počítač. První kapitola práce obsahuje analýzu požadavků na aplikaci a nalezení nejvhodnějšího řešení. V rámci analýzy požadavků nalezneme příklad situace využití aplikace, co se od aplikace požaduje a jaké funkcionality by měla obsahovat. Dále zde také nalezneme technologie, které by měly být pro aplikaci využity. Druhá kapitola práce zmiňuje již v analýze vybrané technologie, které jsou v aplikaci použity a jejich popis, popřípadě rozdíly oproti jiným technologiím. Protože hlavní funkcionalitou aplikace pro uživatele je mapa a práce s ní, je podrobněji popsán výběr a využití daného API. Třetí kapitola práce zmiňuje konkrétní implementaci aplikace, kdy je podrobně popsán návrh databáze, její tabulky, návrh webováho rozhraní, soubory pro práci s databází a vazby mezi jednotlivými prvky aplikace mezi databází a webovým rozhraním. Dále zde jsou také popsány problémy, které při použití jednotlivých technologií nastaly. Ve čtvrté kapitole najdeme ukázky testování z uživatelského pohledu příslušníka společnosti a také z pohledu administrátora společnosti. Toto testování je zobrazeno jak na počítači tak na mobilním zařízení. Dále je zde popsáno, co z pohledu uživatele bylo aplikaci v rámci uživatelského rozhraní vytknuto a naopak co bylo také kladně hodnoceno.

1 Analýza problému

1.1 Analýza požadavků na aplikaci

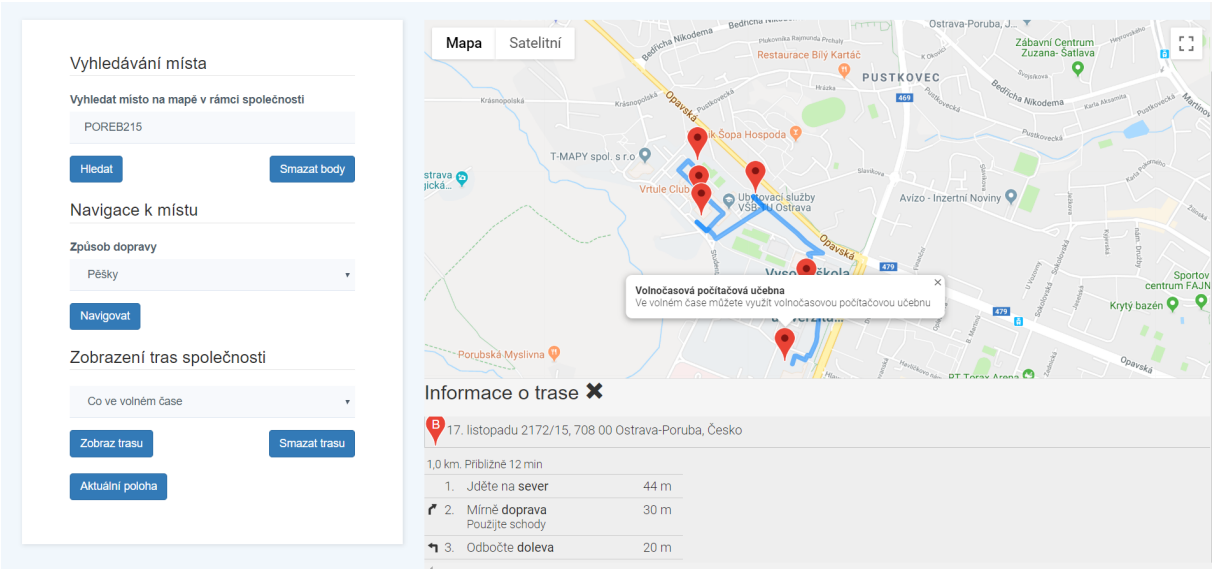
První krok k vytvoření této aplikace byla analýza požadavků na aplikaci. Pro příklad využijeme studenta naší školy. Předpokládejme, že student nastupuje na naši školu a neví, kde má začít a kde se nachází jednotlivé budovy. Z tohoto důvodu využije webové aplikace, která mu po pár kliknutích zobrazí trasu mezi jednotlivými místy, které má navštívit pro vyřízení všech náležitostí. Po kliknutí na jednotlivé místo se studentovi otevře okno na mapě s informacemi, například pro zřízení karty ISIC navštivte kartové oddělení, které je v 1 patře budovy A. Takto může student jednotlivé body proklikat, zjistit o nich informace a navštívit je. Student tedy započal semestr a má zvolený rozvrh. Je první den školy a student v rozvrhu vidí, že jeho první hodina je na učebně POREB204, ale neví, kde ji najít. Napíše tedy do vyhledávání POREB204, dá vyhledat a lokace učebny se mu zobrazí na mapě. Student klikne na tento bod, kdy se mu zobrazí informace o bodu například: Tato učebna se nachází v 2. patře budovy FEI. Takto může vyhledat všechny učebny a budovy, které bude potřebovat navštívit. Skončí škola a student neví, kde by se mohl zajít dobře najíst ve svém okolí. Využije tedy vyhledání zájmového bodu v rámci své polohy a určí si vzdálenost. Po hledání se mu zobrazí body všech restaurací v jeho blízkosti ve vzdálenosti, kterou si určil. Po kliknutí na jednotlivý bod se mu zobrazí informace o vybrané restauraci. Pokud si vybral a neví, jak se k ní dostane, může využít navigaci k dané restauraci s informacemi o trase. Díky této aplikaci pak bude mít student přehled o místech v rámci společnosti a místech v okolí.

1.2 Analýza funkcí aplikace

Druhý krok pro vytvoření aplikace byla podrobná analýza funkcí, které by měla aplikace obsahovat. Aplikace by měla být užitečná v praxi jak pro velké společnosti tak pro malé podniky. Proto, aby se jednotlivé společnosti a jejich uživatelé v aplikaci odlišovali a aplikace byla univerzální, je nutné vytvořit registraci a přihlášení do aplikace, aby se zabránilo neoprávněnému přístupu osoby, která do jednotlivé společnosti či podniku nepatří. Dále by aplikace měla obsahovat správu aplikace pro administrátora společnosti, popřípadě více administrátorů. Pokud by další společnost chtěla vstoupit do aplikace, musela by požádat hlavního administrátora celé aplikace o přidání své společnosti do seznamu společností a databáze. Administrátor společnosti by měl mít možnost své uživatele spravovat, popřípadě vytvářet a spravovat jednotlivé body či trasy v rámci společnosti, aby k nim uživatel mohl využít navigaci. Zaměstnanec či příslušník společnosti by měl mít po registraci a potvrzení administrátorem možnost vyhledat si jednotlivé budovy, místnosti či trasy vytvořené administrátorem v rámci společnosti. Dále by také měl mít možnost využít navigaci v rámci svého okolí, popřípadě vyhledat zájmové body v blízkosti své polohy.

1.3 Analýza volby technologií

Třetím krokem pro vytvoření této aplikace bylo vhodně zvolit technologie, které budou v aplikaci použity. Protože je důležité v dnešní době vytvářet aplikace nezávislé na platformě, je nutné pro webové rozhraní využít technologii, která bude aplikaci zobrazovat v uživatelsky přívětivé podobě a její vzhled bude stejný pro kterýkoliv prohlížeč, velikost zařízení či mobilní zařízení. S tímto problémem souvisí stylování designu aplikace. Z tohoto důvodu bude nutné využít responzivní stylování a takový uživatelsky přívětivý vzhled, který by jak na mobilním zařízení tak na počítači vypadal stejně. Pro vizuální stránku webové aplikace využiji HTML5 a CSS. Na základě provedené analýzy jsem došel k závěru, že pro některé z funkcí aplikace využiji JavaScript. Dále je nutné pro aplikaci využít databázi, která bude obsahovat data uživatelů, společností, jejich tras a bodů. S tímto se váže použití propojení s databází v kódu a využívání dat z databáze v rámci aplikace. Proto bude nutné využití databáze, s ní související propojení a správu dat pomocí PHP. Dále také bude nutné využít mapu. Tím nastává otázka jaké API pro aplikaci využít a jaké bude nejvhodnější. Protože na trhu je velký výběr, bude hlavním cílem využít API, které je dostupné jak pro webovou aplikaci tak pro mobilní aplikaci, protože práce, kterou vytvářím, má i druhou část, která by měla fungovat na platformě Android. Z tohoto důvodu je také nutné využití jednotných systémů a jednotného vzhledu jak pro aplikaci webovou tak pro aplikaci mobilní.



Obrázek 1: Vyhledávání trasy společnosti v prohlížeči Google Chrome

2 Rozbor technologií

2.1 HTML a HTML5

V rámci tvorby webu je základním kamenem jednoduchý textový značkovací jazyk, jenž podporují téměř všechna zařízení - jazyk HTML. Každá webová stránka se zakládá alespoň na minimálním množství kódu jazyka HTML. Tento jazyk byl vynalezen Timem Berners-Leemem, který je současně tvůrcem World Wide Webu, je ředitelem společnosti W3C (World Wide Web Consoritum) a dále pokračuje na jejím vývoji. Jazyk HTML vznikl na počátku 90. let minulého století v podobě stručného dokumentu popisujícího několik elementů používaných pro tvorbu webových stránek. Tyto elementy popisovaly různé části webové stránky jako například odstavce, záhlaví a podobně. Jednotlivé čísla jazyka HTML se zvyšovaly s jejich verzí. Jednotlivé verze byly rozšiřovány o nové prvky a přizpůsobovaly se novým potřebám. Nejnovější verze tohoto jazyka je HTML5. Jazyk HTML5 se vyvinul z přechozích verzí jazyka a snaží se naplnit potřeby současných i budoucích webových stránek. Výhoda jazyka je ta, že pokud jste se naučili jazyk HTML, už znáte většinu jazyka HTML5. Klíčovou vlastností vývoje jazyka HTML5 byla také zpětná kompatibilita se staršími prohlížeči. Jazyk HTML5 také nově zavádí podporu přehrávání zvukových souborů a videosouborů v moderních webových prohlížečích bez jakýchkoliv doplňků [3]. Další důležitou funkcí, kterou HTML5 disponuje je využití JavaScript API, kdy využívá JavaScript API pro geolokaci, která se využívá pro identifikování lokace uživatele, který přistupuje na webovou stránku. Pro webovou aplikaci jsem využil jazyk HTML5, který má mnoho předností oproti jazyku HTML už jen například kvůli výše zmíněné geolokaci.

2.2 CSS a CSS3

Cascading Style Sheets zkráceně CSS je obecně zápis, který určuje vzhled webové stránky. První verze jazyka CSS se objevily až několik let pro vzniku jazyka HTML, kdy oficiálně se tento jazyk prosadil v roce 1996. Jazyk CSS jako jazyk HTML obsahuje několik verzí, kdy s každou novou verzí byly přidány nové funkcionality. Jazyk CSS3 je nejnovější verzí. Zavádí několik nových vizuálních efektů. [3] Jednou z revolučních funkcí jazyka CSS3 je využívání Media Queries Modulu. Tyto dotazy rozšiřují typy médií a dovolují poskytnout konkrétnější styly uživatelského rozhraní, něž jak tomu bylo u dřívějších verzí CSS. Media Queries umožňují, aby byl vizuální vzhled webu opravdu nezávislý na použitém zařízení, a poskytují uživateli nejlepší možný zážitek z webu bez ohledu na to, na jakém zařízení si webovou stránku otevře. W3C dále doporučuje modul Media Queries, tudíž je považován za standard. Tento modul je implementován ve všech hlavních prohlížečích včetně Internet Exploreru od verze 9 [4]. Protože je důležité, aby aplikace byla responzivní bez ohledu na použitém zařízení, využívám v aplikaci CSS3.

2.3 Vizuální framework

Většina webových stránek využívá pro svůj vzhled stránky volně dostupné frameworky. Jedny z nejpoužívanějších jsou aktuálně **Bootstrap** a **Foundation**. Pro svou aplikaci využívám Bootstrap framework z důvodu pro mě osobně lepší dokumentace, zdrojů informací, nástrojů a také lepší kompatibility v rámci starších prohlížečů. Něco málo o technologiích je rozepsáno níže.

2.3.1 Bootstrap

Bootstrap vytvořil Mark Otto a Jacob Thornton pro Twitter v roce 2010, kdy hlavní myšlenkou pro vytvoření Boostrapu byl problém nekonzistence různých aplikací ve firmě, kdy jejich vzhled byl odlišný a byla nutná znalost konkrétního stylu k jejich úpravě. Proto vytvořili tento univerzální framework, který později uvolnili jako open-source, tedy je k dispozici zdarma i pro komerční účely [6]. Jako u HTML i CSS Bootstrap obsahuje několik verzí, kdy s každou verzí přibývají, či ubývají některé funkce tohoto frameworku. Framework využívá prvky HTML, CSS a Javascriptu. Bootstrap je kompatibilní se všemi hlavními prohlížeči a přizpůsobuje se i na starších prohlížečích jako například Internet Explorer 8. Od verze 2.0 také podporuje responzivní design [7]. Tento framework usnadňuje práci s tvorbou rozhraní, vytvářením elementů a zároveň ošetřuje zobrazování napříč platformami [8]. Styly jsou dokonale přizpůsobené pro mobilní zařízení, kdy v dnešní době jsou mobilní zařízení častěji využívané než stolní počítače.

2.3.2 Foundation

Foundation je jako Bootstrap responzivní framework pro vizuální stránku webové aplikace. Obsahuje jednotlivé elementy HTML a CSS pro stylování stránky a také užitečné funkce JavaScriptu. Foundation je také jako Bootstrap open-source projekt. Foundation byl vytvořen společností ZURB a jejím hlavním motivem pro vytvoření tohoto frameworku bylo umožnění vytvářet front-end stránky rychleji a lépe. Od verze 2.0 podporuje responzivní design [9]. Tento framework pro svůj web využívá mnoho známých společností jako například Adobe, Samsung či National Geographic.

2.4 JavaScript

Javascript je multiplatformní, objektově orientovaný skriptovací jazyk, který se zpravidla používá jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky [1]. JavaScript je základním jazykem vývoje webových aplikací. Ať už se webové stránce přidává interaktivita popřípadě se vytváří celé aplikace, dnešní web by nebyl bez JavaScriptu tím, čím je [2]. Jeho autorem je Brendan Eich z tehdejší společnosti Netscape. Syntaxe jazyka patří do rodiny jazyků C, C++, Java [11]. Javascript jde v rámci syntaxe připodobnit k PHP. Pomocí JavaScriptu jsou obvykle ovládány různé interaktivní prvky jako například tlačítka, rozbalovací tabulky, formuláře nebo také animace a efekty obrázků. JavaScript se také

zaměřuje například na ověřování obsahu formulářů na straně klienta. JavaScript funguje na straně klienta narozdíl od například PHP, který funguje na straně serveru. To znamená, že jednotlivé scripty jsou po zobrazení zdrojového kódu čitelné a tudíž s tím souvisí i jistá bezpečnostní omezení, kdy například nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele. [11] JavaScript je velmi silný nástroj pro jednotlivé prvky GUI.

2.5 PHP

PHP - původně Personal Home Page, nyní obvykle rekurzivně Hypertext Preprocessor je skriptovací programovací jazyk určený především pro programování dynamických internetových stránek. Je základním jazykem vývoje webových aplikací. Nejčastěji se začleňuje přímo do struktury jazyka HTML či XML což lze využít při tvorbě webových aplikací. PHP je jedním z nejrozšířenějších skriptovacích jazyků pro web. Oblíbeným se stal díky své jednoduchosti použití, bohaté zásobě funkcí, a tomu, že kombinuje vlastnosti více jazyků a nechává tak vývojáři částečnou svobodu v syntaxi [1]. Při použití PHP jsou skripty prováděny na straně serveru a k uživateli je přenášen jen výsledek jejich činnosti. Uživatel tudíž nemá možnost přistupovat k databázi popřípadě zobrazovat jednotlivé skripty, které jsou začleněny v kódu. Syntaxe jazyka je inspirována jazyky C, Perl, Pascal a Java. Tento jazyk je platformě nezávislý. PHP podporuje celou řadu internetových protokolů jako například HTTP, SMTP, FTP apod. Dále podporuje mnoho knihoven pro různé účely jako je přístup k většině databázových systémů (MySQL, Oracle, ODBC). PHP má velkou podporu na hostingových službách, kdy ho najdeme téměř všude [10].

2.6 MySQL

MySQL je multiplatformní databáze. Pro komunikaci s ní se využívá jazyk SQL. V rámci programování dynamických webových stránek jde o nejpoužívanější databázový systém na světě. Většina webhostingů podporujících jazyk PHP nabízí také databázi MySQL, kterou lze v součinnosti s PHP snadno propojit do podoby dynamické webové aplikace. Databázi MySQL je možné také provozovat na vlastním počítači a vytvořit si tak databázový server [1]. Jednoduchým nástrojem pro správu obsahu databáze MySQL pomocí webového rozhraní je například phpMyAdmin, který obsahují některé hostingy. Jedná se o jeden z nejpopulárnějších nástrojů pro správu databáze [12]. Databáze se skládá z tabulek, které mohou obsahovat data. Tyto data musí být daného datového typu, který se určí při vytváření tabulek. Práce s těmito daty se provádí pomocí CRUD dotazů. Databáze nabízí několik datových úložišť "engine"v závislosti na volbě vývojáře. Engine je motor, který využívá databáze. Nejznámější a nejpoužívanější jsou InnoDB a MyISAM.

2.6.1 MyISAM

MyISAM engine je až do verze 5.1 výchozí formát úložiště dat (storage engine) v databázovém systému MySQL. Využívá full-textové indexování. Tento engine se často používá pro uložení

velkého množství dat a také pro rychlé vyhledávání. MyISAM Nepodporuje cizí klíče a transakce. Používá uzamykání celých tabulek při provádění dotazů. V této době MyISAM stagnuje a všechny budoucí vylepšení by měly být už jen pro InnoDB [14, 15, 16].

2.6.2 InnoDB

InnoDB je jeden z několika formátů úložiště dat (storage engine) v databázovém systému MySQL. InnoDB byl navržen pro zpracování transakcí. Podporuje cizí klíče, které umožňují databázi zajistit integritu a vztahy mezi jednotlivými tabulkami. Používá uzamykání na úrovni řádků a ne nad celými tabulkami, jak je tomu u MyISAM. InnoDB je dále odolnější vůči korupci tabulek než MyISAM. InnoDB nepodporuje full-textové indexování do verze MySQL 5.6. Pro aplikaci jsem si vybral InnoDB jelikož v dnešní době je standardem a má podporu cizích klíčů [13, 15, 16].

2.7 Webhosting

Webhosting je pronájem prostoru pro webové stránky na cizím serveru. Díky webhostingu je možné umístit webové stránky na internet bez nutnosti mít vlastní server. Většina webhostingů je zpoplatněna. Existují i zdarma dostupné webhostingy jako například Endora.cz. Tyto webhostingy zdarma většinou ale vkládají do webových stránek reklamy, mají omezenou rychlost a nemají záruku funkčnosti. Webhosting je pouze samotné umístění stránek na serveru poskytovatele. Pro zpřístupnění stránek uživatelům internetu je nutné mít registrovanou doménu. Tuto doménu lze alternativně získat zdarma u poskytovatele hostingu, ale bývá to doména 3. řádu jako například **nazevwebu.poskytovatel.cz** [1]. Pokud by zákazník vyžadoval doménu druhého řádu, může si za ni zaplatit. Někteří poskytovatelé webhostingu dále nabízí služby jako například generování zabezpečeného certifikátu pro webovou stránku. Pokud bychom chtěli provozovat stránky nad protokolem HTTPS, je nutné mít tento certifikát. V případě některých funkcí, které se využívají v rámci webových aplikací, je přímo HTTPS protokol vyžadován.

2.7.1 Endora

Já jsem si vybral pro svou webovou aplikaci webhosting Endora.cz, protože s ním mám velmi dobré zkušenosti. Tento hosting má stabilní připojení a dobré zabezpečení a umístění serverů. Tento webhosting využívají desítky tisíc webů. Endora nabízí několik variant webhostingu. Nabízí webhosting zdarma pro testovací projekty, kdy je požadována aktivita webu, je zde méně místa pro web a dále je zde vložená reklama. Já jsem si pro svůj projekt vybral webhosting z programu Plus, u kterého je cena 14 Kč/měsíc. Tento program obsahuje větší místo pro web a data jsou uložena na SSD discích. Podporuje dále HTTPS s platným SSL certifikátem zdarma, obsahuje emailovou schránku a podporuje InnoDB engine v databázovém systému MySQL. Pro SSL certifikát Endora využívá nabídky neziskové společnosti Let's Encrypt a nabízí tedy

zdarma zabezpečení komunikace mezi návštěvníkem stránek a serverem pomocí šifrovaného spojení [17, 18].

2.8 Mapy

Výběr API pro mapy je jedním ze základních stavebních kamenů aplikace. Vzhledem k tomu, že aplikace běží jak na mobilním zařízení v systému Android v rámci druhé části práce a zároveň běží jako webová aplikace, je nutné využít API, které je dostupné jak pro web tak pro Android. Služeb pro využití map je dnes mnoho a z tohoto důvodu jsem vybral ty nejpoužívanější. Níže je rozpis průzkumu několika jednotlivých technologií.

2.8.1 Mapy API

Mapy API je provozována společností Seznam.cz a.s. a umožňuje uživatelům používat mapové funkce na vlastních webových stránkách. Použití Mapy API je zcela zdarma a je možné jej využít i pro komerční účely. API má dobře zpracovanou dokumentaci a má výborně zpracovanou mapu České Republiky. Obsahuje většinu prvků, které jsou v dnešní době standardem u konkurence jako například plánování tras, přepínání vrstev mapy, vyhledávání objektů. Základní mapa se aktualizuje obvykle 1x týdně pro Českou republiku a Slovensko a 1x měsíčně pro zbytek světa. Bohužel toto API je zatím dostupné pouze pro web. Neobsahuje SDK pro systém Android, a tudíž by jej nebylo možné použít pro mobilní aplikaci [19, 20].

2.8.2 Leaflet

Leaflet je přední open-source JavaScriptová knihovna pro zobrazení interaktivních map. Obsahuje většinu funkcí, které běžný vývojář potřebuje. Knihovna je navržena s ohledem na výkon, rychlost a jednoduchost. Funguje spolehlivě napříč všemi platformami pro stolní počítače a mobilní zařízení. Podporuje spoustu pluginů, má dobře zpracovanou dokumentaci a jednoduchý a dobře čitelný zdrojový kód. Je velmi oblíbený mezi open-source projekty. Leaflet jako takový bohužel nenabízí vlastní služby jako například Google (geolokace, automatické dopňování zadávání, doprava), ale spoléhá se na služby třetích stran. Z tohoto důvodu může být o něco složitější implementovat tyto služby a ověřit jejich kvalitu. Leaflet nabízí uživatelům různé vrstvy zobrazení, protože se spoléhá na dodavatele třetích stran. Leaflet jako takový není přímo mapovací služba ale pouze knihovna JavaScriptu. Nenabízí tedy ani SDK pro mobilní zařízení. Tato služba se hodí pro menší projekty, které například zobrazují body na mapě nebo mají jednoduché funkce. Pro složitější projekty, jako například navigaci, zobrazení dopravních informací, zpracovávání více dat, nabízení automatického doplňování, se ale nehodí [21, 22, 23].

2.8.3 Google Maps API

Google Maps je platforma od firmy Google, která poskytuje vývojářům rozsáhlé možnosti práce s mapou. Vývojáři mají možnost využít jak SDK pro Android či iOS, tak načítat mapy na

webových stránkách. Mapy pokrývají téměř celé území zeměkoule a snaží se, aby data byla co nejaktuálnější. Google Maps API nabízí hned několik možností rozšíření mapy, kdy má vývojář možnost si vybrat některé z nich a také je kombinovat v závislosti na požadavcích na aplikaci. Některé z nich jsou například:

1. Mapy
    - Mapy JavaScript API
    - Mapy SDK pro Android
    - Mapy SDK pro iOS
    - API pro statické mapy
2. Cesty
    - API pro navigaci
    - API pro silnice
3. Místa
    - API pro místa
    - SDK míst pro Android
    - Geolokace API

Google Maps API v rámci stolních počítačů je podporováno, na všech verzích prohlížečů Chrome a Firefox pro Windows, macOS a Linux. Dále na všech verzích Safari pro macOS a na nynější verzi Microsoft Edge pro Windows. Podporuje také Internet Explorer ve verzích 10 a 11 pro Windows. Pro mobilní systémy je podporován, v prohlížeči Chrome od verze Android 4.1 a pro iOS je podporován, v Safari a Chrome.

Aby vývojář mohl využívat Google Maps API, musí si vytvořit účet a přidat platební informace. Google nabízí zdarma 200 dolarů na měsíc pro využívání jednotlivých funkcí map. Každá služba počítá počet načtení nebo počet volání dané funkce a od toho se pak odvíjí výsledná cena. Pokud by aplikace byla populární a překročila tento limit, bude uživatel vyzván k zaplacení služeb, které překročily limit 200 dolarů. Pokud by tak vývojář neučinil, byly by funkce map nedostupné do dalšího měsíce, kdy by se opět limit 200 dolarů obnovil. Po založení účtu si vývojář vytvoří projekt pro svou aplikaci a k využívání map dostane přidělen API klíč, který poté přidá do načítání map v kódu. Mapy se nenačtou, pokud klíč není platný. Google poskytuje ke svým mapám rozsáhlou dokumentaci a podrobné návody. Některé funkce map vyžadují pro webovou aplikaci protokol HTTPS, jinak její funkce nebudou dostupné. Jedna z těchto funkcí je například Geolokace. Z tohoto důvodu je nutné mít na stránkách ověřený důvěryhodný certifikát. Z důvodů podpory, API pro web, SDK pro Android, volně dostupných funkcí do výše 200 dolarů a výborně zpracované dokumentace jsme se rozhodli využít tyto mapy pro webovou a mobilní aplikaci [24].

3 Implementace technologií a tvorba aplikace

Díky předešlé analýze požadavků a volby jednotlivých technologií je nyní možné aplikaci realizovat. V této části práce je popsán popis implementace jednotlivých technologií a tvorba aplikace.

3.1 Tvorba databáze

Prvním krokem k vytvoření aplikace je návrh a tvorba databáze, protože z analýzy požadavků víme, že aplikace bude využívat databázi pro uchovávání dat a jejich správu. Databázi jsem vytvořil na webhostingu Endora.cz. Pro databázi využívám MySQL engine InnoDB. Databáze je rozdělena na několik tabulek, které v sobě uchovávají data. Níže je podrobný popis všech tabulek databáze.

3.1.1 Tabulka Corp

Protože je aplikace stavěná pro podniky a společnosti, vytvořil jsem si tabulku Corp, která reprezentuje jednotlivé společnosti. Každá společnosti má svůj identifikátor datového typu int, který je zároven i primárním klíčem tabulky a podle něj se přidělují ostatní uživatelé, kteří se do aplikace registrují. Každá společnost má atributy Name a City typu varchar, které reprezentují název a město společnosti. Poslední dva atributy, které tabulka obsahuje jsou Longitude a Latitude datového typu float. Tyto atributy reprezentují zeměpisnou délku a zeměpisnou šířku společnosti v rámci mapy.

3.1.2 Tabulka Users

Další tabulkou databáze je tabulka reprezentující uživatele. Každý uživatel má svůj identifikátor datového typu int a je také primárním klíčem. Dále má uživatel atributy FirstName, LastName, City datového typu varchar, které reprezentují jméno, příjmení a město uživatele. Jelikož se uživatel do aplikace registruje a poté se musí přihlásit, obsahuje tabulka atributy Login a Pass, kdy Login i Pass je typu varchar a reprezentují přihlašovací jméno a heslo uživatele. Heslo je uložené v databázi v šifrované podobě. Dále tabulka obsahuje atribut Mail typu varchar, který slouží v případě zapomenutého hesla k odeslání na zadaný e-mail. Další atribut tabulky je Admin typu tinyint. Tento atribut znázorňuje úroveň uživatele aplikace. Rozlišuje se mezi hodnotou 0 a 1, což definuje oprávnění uživatele, viz sekce 3.2.1. Další z atributů je Approved typu tinyint, který určuje, zda se uživatel do aplikace může přihlásit nebo ne. Aby se uživatel do aplikace mohl přihlásit, je nutné, aby ho správce aktivoval a tím mu nastavil hodnotu Approved na 1. V případě, že by správce chtěl uživatele deaktivovat, nastaví mu hodnotu Approved na 0. Atributy Approved a Admin mají výchozí hodnotu 0. Poslední z atributů tabulky je CorpID, který je datového typu int a zároveň je také cizím klíčem, který odkazuje na tabulku Corp. Tímto zajišťuji, aby se uživatel nemohl registrovat do neexistující společnosti a pokud by se společnost

odstranila, je možné z ní odstranit i všechny její uživatele. Tímto atributem tedy určuji, do jaké společnosti daný uživatel patří. Všechny atributy jsou povinné.

3.1.3 Tabulka Pass_reset

Tato tabulka je určena pro odeslání zapomenutého hesla uživateli. Obsahuje atribut UserID datového typu int, který je také cizí klíč, odkazující na tabulku users. Dále obsahuje atribut keyg datového typu varchar, který reprezentuje klíč, který se generuje při odeslání žádosti o zapomenuté heslo. Dále také obsahuje atribut expDate datového typu datetime, který reprezentuje datum expirace klíče. Po resetování hesla se takovýto vytvořený záznam z databáze odstraní. Všechny atributy jsou povinné.

3.1.4 Tabulka Mark

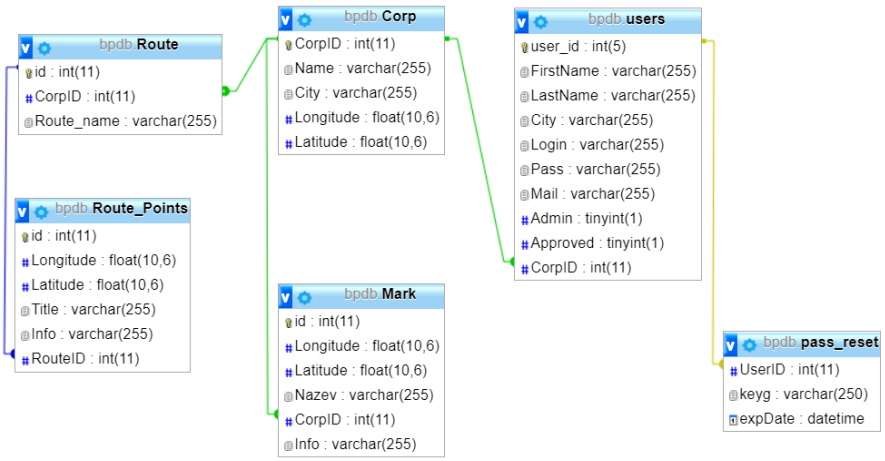
Protože správce společnosti má možnost vytvářet místa, která se následně mohou zobrazit na mapě, vytvořil jsem si tabulku Mark. Tato tabulka obsahuje atribut id, který je identifikátor místa a zároveň primárním klíčem. Dále obsahuje atributy Longitude a Latitude datového typu float, které určují zeměpisnou délku a šířku daného místa. Dále jsou zde atributy Nazev a Info datového typu varchar, které reprezentují název místa a informaci o místu. Atribut Info je nepovinný. Poslední z atributů je CorpID datového typu int, který je zároveň cizím klíčem, který odkazuje na tabulku Corp. Tento atribut určuje místo, které patří dané společnosti. Všechny atributy kromě atributu Info jsou povinné.

3.1.5 Tabulka Route

Aplikace umožňuje správci přidávat cesty, které obsahují své body a následně se po výběru cesty zobrazí na mapě. Z tohoto důvodu jsem vytvořil tabulku Route, která má atribut id typu int, který je primární klíč a také identifikátor jednotlivé cesty. Dále zde je atribut Route_name typu varchar, který reprezentuje název dané cesty. Poslední z atributů je CorpID typu int, který je zároveň cizím klíčem a odkazuje na tabulku Corp. Tímto je zajištěno, aby každá cesta patřila do nějaké společnosti v závislosti na identifikátoru společnosti. Všechny atributy jsou povinné.

3.1.6 Tabulka Route_Points

Tato tabulka reprezentuje jednotlivé body vytvořených cest. Obsahuje atribut id typu int, který je identifikátorem a zároveň primárním klíčem. Dále obsahuje atributy Longitude a Latitude typu float, které reprezentují zeměpisnou délku a šířku daného bodu. Dále obsahuje atribut Title a Info typu varchar určující název místa a informaci o daném místu. Atribut Info je nepovinný. Poslední z atributů je RouteID který je typu int a zároveň je cizím klíčem odkazující na tabulku Route. Tímto je zajištěno, že dané místo patří dané cestě a zároveň je tedy zajištěno, že místo, které patří do cesty, patří zároveň do dané společnosti. Všechny atributy jsou povinné až na vyjímku atributu Info.



Obrázek 2: Datový model

3.2 Tvorba vzhledu webové aplikace

Druhým krokem pro vytvoření aplikace je tvorba webové stránky. Pro vytvoření stránky jsem využil HTML jazyk a CSS stylování. Dále pro aplikaci používám Bootstrap framework pro vytvoření základního stylu stránky a její responzivitě. Jedním z důležitých faktorů pro dobrou webovou aplikaci je její přehlednost. Z tohoto důvodu je nutné navrhnout správné rozložení prvků, protože tento faktor je většinou pro uživatele klíčový v rámci orientace v aplikaci.

3.2.1 Menu

Základním prvkem aplikace je menu, přes které se uživatel aplikace orientuje. Toto menu je dále implementováno pro všechny následující stránky. Menu je přizpůsobitelné velikosti zařízení, kdy pokud je šířka zařízení větší než 767 obrazových bodů, je zobrazeno klasické desktopové menu. Pokud má ale zařízení rozlišení menší než 768 obrazových bodů, vytvoří se tlačítko pro rozbalení daného menu. Protože aplikace funguje na principu klasického uživatele, přihlášeného uživatele a administrátora, je důležité vytvořit 3 různé menu, které se budou zobrazovat v závislosti na úrovni přihlášeného uživatele. Protože se menu zobrazuje na všech stránkách, vytvořil jsem si soubory, které poté načítám na každé stránce v závislosti na úrovni uživatele. Načítání daného menu je realizováno pomocí PHP, kdy je kontrolováno, zda uživatel, který do aplikace vstoupil, je přihlášen nebo ne. Pokud uživatel není přihlášen, načítá se menu pro nepřihlášeného uživatele. Pokud uživatel je přihlášen, tak je kontrolováno, zda má nastavenou hodnotu Admin na 1 nebo 0, a podle toho je načteno příslušné menu. Jednotlivé menu jsem pojmenoval podle úrovně uživatele. Tímto jsem vytvořil 3 různé soubory pro menu. Každý soubor má ve svém menu jiné prvky.

- menu_neprihlasen.php
  - Nepřihlášený uživatel vidí v aplikaci pouze tlačítko domů, tlačítko registrace a rozbalovací lištu pro přihlášení.
- menu_prihlasen.php
  - Přihlášený uživatel, který má v databázi nastavenou hodnotu v tabulce Admin na 0, vidí v menu odkaz pro mapy, místa společnosti, registraci a rozbalovací informaci o sobě s tlačítkem pro odhlášení.
- menu_admin.php
  - Správce má již rozsáhlejší menu, a to z důvodu administrace aplikace. V tabulce Admin má v databázi nastavenou hodonotu na 1. Jeho menu obsahuje proto tlačítka pro přidání destinace, úpravu destinace, přidání trasy, místa tras, mapy, místa společnosti, aktivaci uživatelů, registraci a rozbalovací informaci o sobě s tlačítkem pro odhlášení.

3.2.2 Domovská stránka

Domovská stránka, která se nachází v souboru index.php, obsahuje informace o aplikaci, aby uživatel, který stránky navštíví, mohl vidět, co aplikace umí a obsahuje. Tyto informace uživatel najde v přehledných dlaždicích s ikonou a popiskem, o obsahu informace. Bootstrap využívá pro rozvržení prvků stránky tabulkový systém. Dále má možnost využívat různé stylování pro zobrazení prvků na mobilním zařízení a na stolním počítači. Další funkcionalitou, kterou Bootstrap disponuje jsou ikony glyphicons. Tyto ikony jsou k využití zdarma a hodí se pro různé druhy popisků, prvků či zobrazení informací. Pro zobrazení jednotlivých kostek s informacemi využívám tabulkový systém rozložení jednotlivých prvků a také již zmíněné glyphicons. Dále se na hlavní stránce načítá výše zmíněné menu pomocí PHP.

3.2.3 Registrace

Stránka pro registraci je vytvořená velmi jednoduše, aby se v ní každý uživatel vyznal. Kód stránky je umístěn v souboru register.php. Registrace obsahuje formulář, který odkazuje po odeslání na PHP soubor registration.php pracující s databází viz sekce 3.3.2. Formulář obsahuje pole pro jméno, příjmení, e-mail, město, přihlašovací jméno, heslo a rozbalovací seznam pro výběr společnosti, do které se uživatel registruje. Všechny tyto atributy jsou povinné. Každému poli je přiřazena ikona z glyphicons a nápověda, co do daného pole patří. Nápověda je řešená pomocí HTML5 atributu placeholder.

3.2.4 Přidání místa

Stránka obsahuje formulář pro přidání místa do databáze. Formulář se skládá z názvu místa, zeměpisné délky daného místa, zeměpisné šířky a informací o místu. Pokud by správce chtěl přidat do databáze například učebnu POREB215, která by se následně uživatelům aplikace zobrazovala ve vyhledávání a mohla se zobrazit do mapy, zadá správce název místa POREB213, zeměpisnou délku 18.160728, zeměpisnou šířku 49.831200 a popřípadě informaci (například učebna se nachází ve 2. patře budovy FEI) a nakonec dá přidat. Tímto se odešle požadavek, který zpracovává soubor insert_dest_db.php, viz sekce 3.3.6. Jelikož stránka je přístupná pouze pro správce, kontroluji pomocí PHP, zda je uživatel přihlášen a má oprávnění správce, viz sekce 3.3.4.

3.2.5 Úprava místa

Stránka je určena k úpravě, popřípadě smazání místa, které bylo přidáno na stránce pro přidání místa, viz sekce 3.2.4. Na stránce se nachází rozbalovací seznam míst, která odpovídají dané společnosti a jsou uloženy v databázi. Pro využívání těchto funkcí má oprávnění pouze správce. Správce má tedy možnost jednotlivé místa smazat, popřípadě upravit jejich názvy nebo souřadnice, pokud například při vkládání došlo k překlepu, či přidat nebo upravit dodatečně jejich popis. Dále se zde nachází formulář pro výše zmínenou úpravu. Jednotlivé prvky stránky jsou zobrazeny po kontrole oprávnění, viz sekce 3.3.4. Pokud uživatel projde kontrolou, odešle se požadavek na databázi, kde se dotazuje na místa patřící dané společnosti, do které je správce zařazen. Po obdržení míst se prochází počet řádků jednotlivých míst a ty se následně v cyklu vypisují do responzivní tabulky. Každému řádku je přiřazeno tlačítko pro smazání, úpravu místa a skrytý identifikátor. Tlačítko pro smazání obsahuje odkaz na soubor del_place_db.php, viz sekce 3.3.7. Vytvořená tabulka se následně skryje do rozbalovacího seznamu, který je realizován pomocí třídy .collapse, která se stará o zobrazení a schování tabulky a patří do komponent Bootstrapu. Pokud správce klikne na tlačítko upravit, data z daného řádku se překopírují do formuláře pro úpravu. Toto je realizováno pomocí JavaScriptu, kdy se do proměnných načtou nejbližší hodnoty v řádku z tabulky a následně se zapíšou do kolonek formuláře. Správce může nyní data přepsat nebo opravit. Po stisknutí tlačítka aktualizovat se odkáže na soubor update_dest_db.php, viz sekce 3.3.7.

3.2.6 Přidání trasy

Tato stránka slouží pro přidání trasy do databáze. Následně správce k těmto trasám může přidat místa a poté si tyto trasy, které byly vytvořeny správcem, uživatelé společnosti mohou zobrazit na mapě. Stránka je opět přístupná pouze pro správce a je tedy prováděna kontrola, viz sekce 3.3.4. Stránka se skládá ze tří sekcí.

První sekce je rozbalovací seznam aktuálních tras v databázi, které si správce může zobrazit po kliknutí na tlačítko zobrazit seznam tras. Tyto trasy může mazat a upravovat jejich název. Seznam je vytvořen pomocí dotazu, který vybírá všechny cesty z databáze na základě podmínky,

kde se identifikátor společnosti rovná identifikátoru společnosti uloženém v $_SESSION uživatele. Poté se názvy tras umístí po řádku do responzivní tabulky a k nim se přidají tlačítka pro úpravu a smazání. Tlačítko pro smazání odkazuje na soubor del_route_db.php a ten následně zajistí vymazání trasy, viz sekce 3.3.8. Tlačítko pro úpravu zkopíruje název trasy a její identifikátor, který je schovaný a vloží je do formuláře pro úpravu názvu trasy, který se nachází ve 3 sekci stránky.

V druhé sekci stránky je umístěn formulář pro přidání trasy do databáze. Správce zadá název trasy, a stiskne tlačítko přidat. O přidání trasy do databáze se stará soubor insert_route_db.php, viz sekce 3.3.8. Po přidání trasy se nově přidaná trasa zobrazí v seznamu tras a ve výběru tras na stránce míst společnosti.

Ve třetí sekci najdeme již zmínění formulář pro úpravu názvu trasy. O tuto úpravu se stará soubor update_route_db.php, viz sekce 3.3.8.

3.2.7 Přidání míst pro trasu

Tato stránka se skládá ze 3 sekcí. Je určena pro přidání a úpravu míst jednotlivých tras. Stránka je určena pouze pro správce, a proto je prováděna kontrola viz sekce 3.3.4.

V první sekci je formulář pro přidání místa k trase. Skládá se z rozbalovací nabídky pro výběr trasy, názvu místa, zeměpisné délky, zeměpisné šířky a informací o místu. Všechny hodnoty kromě informace o místu jsou povinné. Výpis rozbalovací nabídky s trasami jsem provedl pomocí dotazu, který vypíše všechny cesty z databáze, kde identifikátor společnosti je roven identifikátoru společnosti, který je uložen v $_SESSION uživatele. Tímto je zaručeno, že se správci vypíšou pouze trasy, které patří jeho společnosti. Rozbalovací seznam je HTML element option, který má parametr value, do kterého ukládám identifikátor dané cesty. Správce má tedy možnost si vybrat, do které trasy chce přidat nové místo, které se poté bude zobrazovat po výběru trasy na mapě. Po vyplnění všech ostatních polí a odeslání požadavku se o přidání stará soubor insert_route_point_db.php, viz sekce 3.3.9.

V druhé sekci je aktuální seznam tras a jejich míst. Jsou zde vypsány všechny trasy, které patří společnosti správce. Tyto trasy jsou vypsány pomocí dotazu, který byl popsán v přidání místa pro trasu. Pro přiřazení jednotlivých míst, které odpovídají daným trasám, jsem si vytvořil druhý dotaz, který spojí tabulku Route a Route_Points na základě id z tabulky Route a RouteID z tabulky Route_Points, kde identifikátor společnosti je roven identifikátoru společnosti uloženém v $_SESSION uživatele. Toto propojení provádím proto, že potřebuji identifikátor společnosti, který se v tabulce Route_Points nenachází. Po vykonání dotaz vybere z tabulky Route_Points informace, které chci zobrazit a uloží je do proměnné. Díky předchozím dvěma dotazům lze provést rozdělení míst k trasám. V cyklu si výsledek druhého dotazu pomocí funkce fetch_assoc() ukládám do proměné jako asociativní pole. Po uložení kontroluji podmínku, kde kontroluji, zda se identifikátor trasy z prvního dotazu rovná identifikátoru trasy z druhého dotazu. Pokud ano, jsou vypsány dané informace z druhého dotazu do tabulky a s nimi i tlačítko pro úpravu a smazání místa trasy. Tímto se vypsala celá tabulka. O smazání trasy se stará sou-

bor del_route_point_db.php, viz sekce 3.3.9. Pokud by správce chtěl upravit informace o místu trasy, může využít tlačítka upravit, které pomocí JavaScriptu rozkopíruje data do jednotlivých polí formuláře, viz sekce 3.2.5, který se nachází v 3 sekci stránky.

V třetí sekci stránky je umístěn formulář pro úpravu hodnot místa trasy. Všechny parametry formuláře jsou povinné až na informaci o místu. O úpravu informací se stará soubor update_route_point_db.php, viz sekce 3.3.9.

3.2.8 Správa uživatelů

Tato stránka slouží ke správě uživatelů společnosti. Stránka je dostupná pouze pro správce, a proto probíhá jeho ověření, viz sekce 3.3.4. Správce má možnost uživatele, kteří se zaregistrovali do jeho společnosti, aktivovat či deaktivovat. Pokud uživatel není aktivován, nemá možnost se přihlásit. Pokud se uživatel pokusí přihlásit a není aktivovaný, zobrazí se mu hláška s informací, že jeho účet není aktivován. Stránka je rozdělena do dvou částí. V první části je okno s responzivní tabulkou, ve které je seznam uživatelů čekajících na potvrzení. Ke každému uživateli je na řádku vypsáno jméno, příjmení, město a tlačítko pro jeho aktivaci. O aktivaci uživatele se po stisknutí tlačítka stará soubor update_users_db.php, který je podrobněji popsán v sekci 3.3.10. V druhé části stránky správce najde okno pro deaktivaci uživatelů. Zde je do tabulky vypsán seznam aktuálně aktivovaných uživatelů s jejich informacemi a tlačítkem pro deaktivování. O deaktivaci uživatele se stará soubor update_users_db.php, viz sekce 3.3.10. Jednotlivý výpis uživatelů jak pro aktivaci tak deaktivaci, provádím pomocí dvou dotazů. První dotaz vybere z tabulky users identifikátor uživatele, jméno, příjmení a město, kde hodnota v sloupci Approved je nastavena na 0, a zároveň kde identifikátor společnosti je roven identifikátoru společnosti uloženém v $_SESSION uživatele. Následně pak tento výsledek ukládám do proměnné a pomocí funkce fetch_assoc() ho v cyklu jako asociativní pole přiřazuji do nové proměnné. V cyklu poté vypisuji jednotlivé řádky tabulky s hodnotami z dotazu pro každého uživatele společnosti. Druhý dotaz vybere z tabulky users identifikátor uživatele, jeho jméno, příjmení a město, kde hodnota v sloupci Approved je nastavena na 1 a zároveň je identifikátor společnosti roven identifikátoru společnosti uloženém v $_SESSION uživatele. Následně probíhá výpis uživatelů jako v předchozím dotazu a každému řádku je přidělen příslušný identifikátor uživatele a parametry pro deaktivování s hodnotou 0 do skrytého elementu.

3.2.9 Mapy

Tato stránka umožňuje přihlášenému uživateli orientaci a vyhledání zájmových míst v jeho okolí. Dále také nabízí možnost navigace k jednotlivým bodům a zobrazení informací o cestě. Uživatel si také může zvolit způsob dopravy, který chce použít. Stránka je rozdělena na 2 části. Vlevo uživatel najde panel pro práci s mapou a napravo mapu. Tato mapa je na stránce vložena pomocí JavaScriptu, ve kterém je odkaz na Google Maps API, kde je v parametru odkazu vložený API key, knihovny a odkaz na funkci načítající mapu, viz sekce 3.4. Levý panel se skládá ze tří

částí, které jsou oddělené pomocí legendy. První je vyhledání místa na mapě, kde má uživatel možnost vyhledat místo, které zadá. Při zadávání se uživateli zobrazují návrhy míst. Při vybrání daného místa se uživateli na mapě zobrazí značka místa, které vyhledal a mapa se zaměří na danou značku. Po kliknutí na značku se uživateli zobrazí informace o daném místu. V druhé části panelu je vyhledávání zájmových míst v okolí aktuální polohy. Vyhledávání v okolí je v jednotkách metrů. Uživatel do pole tedy zadá rozsah okolí v metrech, ve kterém chce zobrazit zájmová místa kolem své aktuální polohy. Uživatel si může pomocí rozbalovacího seznamu vybrat jednotlivé kategorie typů míst. Po stisknutí tlačítka hledat se mu zobrazí značky míst v rozsahu zadaného okolí v metrech. Po kliknutí na jednotlivé značky si může zobrazit informace o místech, popřípadě k nim využít navigaci. Pokud uživatel chce značky smazat, slouží k tomu tlačítko smazat body. V poslední části najde uživatel způsob navigace a tlačítko pro navigování. Pokud si uživatel zobrazí navigaci k místu a chce poté trasu navigace z mapy vymazat, může využít tlačítko smazat trasu. Jako poslední prvek v levém panelu je tlačítko aktuální poloha, která vyzve uživatele o povolení aktuální polohy, pokud tak již neučinil při načtení stránky, kdy pomocí HTML5 geolokace je uživatel vyzván k povolení určení polohy, a následně se zobrazí aktuální poloha uživatele na mapě. Stránka je stylována tak, aby uživateli nabídla co nejlepší přehlednost. Panel je nastaven na 35% šířky stránky na zařízení uživatele a mapa je nastavena na 65% šířky stránky. Pokud by uživatel otevřel stránku na zařízení, které má šířku obrazových bodů menší než 737 obrazových bodů, zobrazí se mu panel nad mapou. Toto řešení jsem využil, aby i v případě využití aplikace na mobilním zařízení stránka vypadala přívětivě. Toto zobrazení dociluji pomocí CSS media query.

3.2.10 Místa společnosti

Tato stránka je určena především pro zobrazení míst společnosti, které správce společnosti přidal do databáze. Dále zde má také možnost využití navigace k jednotlivým místům a zobrazení tras společnosti. Na stránce využívám pro načtení mapy HTML5 <script> element, který obsahuje odkaz na Google Maps API, kde parametry odkazu jsou API klíč, potřebné knihovny a odkaz na funkci načítající mapu, viz sekce 3.4.1. Stránka je rozdělena do tří částí formou legendy. V první části je pole pro vyhledávání místa společnosti. Pokud uživatel hledá místo společnosti, zadá její název, a pokud dané místo je již správcem vytvořeno automaticky, zobrazí se mu místo pro doplnění popřípadě další místa, která mají podobný název. Po vybrání daného místa a stisknutí tlačítka hledat se uživateli na mapě zobrazí značka místa. Pokud na vybranou značku uživatel klikne, zobrazí se mu informace o místu, pokud nějakou informaci správce k místu přidal, a název daného místa. Uživatel následně může využít navigaci k danému místu pomocí navigace, která se nachází v druhé části panelu. Pokud již bod na mapě nepotřebuje, nebo si chce vymazat bod z mapy, může využít tlačítko smazat body, které vymaže všechny vyhledané body na mapě s výjimkou bodu aktuální polohy a bodu určující lokaci společnosti. V druhé části panelu je výběr způsobu dopravy a tlačítko pro navigaci. Zde si uživatel může vybrat způsob dopravy, který chce použít. Ve třetí části stránky se nachází zobrazení tras společnosti. V této části má

uživatel možnost vybrat si z předpřipravených tras správcem společnosti a zobrazit si tak body dané trasy na mapě. Po kliknutí na jednotlivé body se zobrazí název bodu a informace k němu, pokud ji správce k místu trasy přidal. Pro smazání trasy může využít tlačítko smazat trasu. Poslední prvek stránky je tlačítko pro určení aktuální polohy. Stylování stránky a geolokace jsou stejné jako v sekci 3.2.9.

3.3 Soubory pro práci s databází

3.3.1 Kontrola přijatých informací z formuláře

V rámci průzkumu nejzávažnějších zranitelností webových aplikací [25] jsem narazil na problém, při kterém může útočník neoprávněně zasahovat do databáze přes neošetřené vstupy. Jedná se například o SQL Injection, což je technika napadení databázové vrstvy programu vsunutím kódu přes neošetřený vstup, viz [5]. Proto v každém souboru, který přijmá a následně zpracovává data z formuláře, využívám funkci mysqli_real_escape_string(), která přidá znakům, které by mohly zapříčinit změnu v databázi, zpětné lomínko.

3.3.2 Registrace uživatele

Registraci uživatele do aplikace zpracovává soubor registration.php. Po přijetí uživatelských informací z formuláře tyto hodnoty přiřazuji do proměnných a kontroluji, zda hodnoty nejsou prázdné. Protože heslo do databáze je třeba ukládat v zabezpečené podobě, používám PHP metodu password_hash(), která slouží k bezpečnému vytváření hashů pro hesla. Do této metody předávám proměnnou s heslem, která přišla z formuláře a dále konstantu PASSWORD_DEFAULT, která se může časem změnit v závislosti na přidání nových a silnějších algoritmů do PHP. Tato konstanta využívá nejlepší možný algoritmus pro vytvoření hashe v dané verzi PHP. Od verze PHP 5.5.0 se pro PASSWORD_DEFAULT používá jako algoritmus bcrypt. Následně provádím kontrolu e-mailu, z kterého odstraňuji všechny nedovolené znaky a kontroluji, zda je e-mail platný. Další kontrolou, kterou provádím, je zda login a e-mail zadaný při registraci není již v databázi obsažen. Kontrolu provádím dotazem, který z tabulky users vybere přihlašovací jméno, za podmínky, že přihlašovací jméno v databázi se rovná přihlašovacímu jménu z formuláře. Pro kontrolu e-mailu používám dotaz, který vybere z tabulky users e-mail, za podmínky, že e-mail v databázi se rovná e-mailu z formuláře. Pokud je počet řádků, které dotaz vrátil roven jedné, uživateli vypíši informaci, že zadané uživatelské jméno nebo e-mail je již obsazen. Pokud ale dotaz vrátí prázdný výsledek, je možné uživatelské jméno a e-mail použít. Poté přecházím k dotazu, který provede vložení do databáze, kde do tabulky users vkládám hodnoty z formuláře. Pokud vložení proběhlo úspěšně, je uživatel informován o úspěšné registraci a je požádán o počkání na aktivaci účtu správcem.

3.3.3 Přihlášení uživatele

Menu obsahuje rozbalovací formulář pro přihlášení. Po vyplnění přihlašovacích údajů se o přihlášení stará soubor check.php. V tomto souboru kontroluji pomocí podmínek, zda hodnoty, které přišly z formuláře pro přihlášení, nejsou prázdné. Dále zde mám vytvořený dotaz, který vybere z tabulky users identifkátor uživatele, heslo a atribut, zda je uživatel potvrzený či ne, ze sloupce Approved, kde se přihlašovací jméno uživatele rovná jménu z formuláře. Tento výsledek dotazu ukládám do proměnné a ověřuji pomocí podmínky, zda v dané proměnné je alespoň 1 záznam. Pokud by nebyl, znamenalo by to, že dotaz nevrátil žádný výsledek, a tak v databázi neexistuje žádný uživatel s přihlašovacím jménem, který byl zaslán formulářem. Proto uživateli vypíši hlášku, že zadaný uživatel neexistuje. Pokud je ale podmínka splněna, pomocí funkce mysqli_fetch_assoc() uložím do nové proměnné výsledek dotazu jako asociativní pole. Tím pádem mohu s hodnotami, které jsem si vybral v předchozím dotazu, pracovat a ověřit pomocí podmínky, zda danému uživateli odpovídá zadané heslo a je nastaven jako aktivní uživatel. Protože je heslo v databázi v zašifrované formě, využívám funkci password_verify(), která má 2 parametry, heslo, které přišlo z formuláře a heslo, které jsem pomocí dotazu vybral z databáze pro daného uživatele. Tato funkce porovnává heslo s hashem, který jsem pomocí dotazu vybral z databáze. Zároveň v této podmínce ověřuji, zda má uživatel nastaven parametr Approved na 1. Pokud by heslo neodpovídalo přihlašovacímu jménu, uživateli se vypíše hláška, která mu oznámí, že zadané heslo je nesprávné, a pokud by uživatel měl správné heslo, ale nebyl aktivován, vypíše se mu hláška, že jeho uživatelský účet není aktivován. Pokud je podmínka splněna, pomocí dotazu vyberu všechny informace z tabulky users pro daného uživatele, kdy tabulku users spojuji s tabulkou Corp pomocí hodnoty CorpID za podmínky, že identifikátor uživatele je roven identifikátoru uživatele z prvního dotazu. Tyto tabulky spojuji proto, abych si z tabulky Corp mohl pro daného uživatele uložit hodnotu názvu společnosti, která se po přihlášení uživateli zobrazí v jeho profilu, a dále hodnoty zeměpisné šířky a délky společnosti, které poté používám pro stránku místa společnosti. Do $_SESSION uživatele ukládám identifikátor uživatele, jméno, příjmení, e-mail, zda je uživatel správce nebo ne, identifikátor společnosti, zeměpisnou šířku, zeměpisnou délku a název společnosti. Po přihlášení je uživatel přesměrován na domovskou stránku a v menu je nyní zobrazen jeho login a po rozkliknutí je zobrazeno jeho jméno, příjmení, e-mail a název jeho společnosti.

3.3.4 Ověřování uživatele

Jelikož aplikace rozlišuje různé uživatele, kteří mají různá oprávnění, a je tedy nutné toto oprávnění na různých stránkách ověřovat, vytvořil jsem si systém, který rozlišuje, zda uživatel má v databázi nastavenou hodnotu atributu Admin v tabulce users na 1, nebo 0. Tímto v aplikaci ověřuji, co zobrazit danému uživateli. Ověřování probíhá pomocí PHP, kdy se kontroluje $_SESSION uživatele, který je aktuálně přihlášený. Pokud má uživatel ve svém $_SESSION uloženou hodnotu atributu Admin na 1, má povolení zobrazit všechny stránky aplikace, viz sekce

3.2.1. Pokud má uživatel v $_SESSION hodnotu 0, může zobrazit pouze uživatelské odkazy, viz sekce 3.2.1. Pokud by uživatel s hodnotou 0 otevřel stránku, na kterou má přístup pouze správce aplikace, zobrazí se mu hláška s informací, že pro zobrazení této stránky nemá přístup. Pokud by tak učinil uživatel, který není ani přihlášen, zobrazí se mu na stránce hláška s informací, že tyto stránky mohou navštívit jen registrovaní uživatelé. Ověřování oprávnění uživatele dále také používám v jednotlivých souborech pro práci s databází.

3.3.5 Kontrola společnosti a jejich dat

Tuto kontrolu provádím z důvodu, kdyby někdo chtěl ve zdrojovém kódu stránky upravit identifikátor uživatele, trasy nebo míst, který je skrytý, a tím pozměnit, smazat nebo přidat něco v jiné společnosti. V této kontrole se provede dotaz na databázi, který vybere identifikátor z tabulky, pro kterou se provádí dané úpravy na základě podmínky, kdy se porovnává identifikátor z tabulky s identifikátorem, který byl odeslán formou formuláře, a zároveň identifikátor společnosti, který se rovná identifikátoru společnosti uživatele, který se uložil do $_SESSION uživatele při přihlášení. Po této kontrole se v případě úpravy uživatele či místa do proměnné uloží pouze jeden jediný identifikátor, protože dotaz vrátí jeden záznam. Tento identifikátor se poté v podmínce porovnává s identifikátorem, který byl odeslán v rámci formuláře. Pokud se rovnají, provede se dotaz pro úpravu, přidání či smazání místa, či aktivaci nebo deaktivaci uživatele. Pokud dotaz proběhl v pořádku, vypíše se uživateli hláška o úspěšném provedení akce. Pokud se ale identifikátory nerovnají, zobrazí se uživateli chybová hláška, která ho informuje, že nejde měnit, mazat ani přidávat data jiné společnosti, než té, do které patří. Je zde ale několik výjimek, kdy se pouze jeden záznam v rámci dotazu nevrátí.

První výjimka nastává u odebrání trasy, kdy se při provedení dotazu vyberou z databáze všechny identifikátory tras na základě podmínky, kde se porovnává identifikátor společnosti s identifikátorem, který je uložený v $_SESSION uživatele. Poté se prochází v cyklu jednotlivé identifikátory, které byly vybrány v dotazu, a kontroluje se, zda identifikátor, který je zrovna vybrán, je odlišný od identifikátoru, který byl zaslán formulářem. Pokud odlišný je, neprovádí se další kroky a opět se kontroluje další. V případě, že ani jeden identifikátor neodpovídá tomu z formuláře, uživateli se zobrazí chybová hláška, která ho informuje, že nejde mazat trasy společnosti, do které nepatří. Pokud ale identifikátor odeslaný formulářem odpovídá některému z identifikátorů vybraných z dotazu, provede se další dotaz na databázi, který smaže všechny místa dané trasy na základě podmínky, že identifikátor trasy je roven identifikátoru z formuláře. Tento krok je nutné vykonat, protože by databáze nedovolila smazat trasu bez odebrání jejich míst, z důvodu existence cizího klíče. Po odebrání míst trasy se pokračuje smazáním již konkrétní trasy na základě podmínky, že identifikátor trasy je roven identifikátoru z formuláře. V případě, že vše proběhlo úspěšně, vypíše se uživateli hláška o úspěšném smazání.

Druhá výjimka nastává při přidání místa trasy. Pro přidání místa pro danou trasu, musím vybrat z tabulky Route, všechny identifikátory tras, kdy identifikátor společnosti je roven identifikátoru, který je uložený v $_SESSION uživatele. Výsledek dotazu ukládám do proměnné, ze

které následně v cyklu pomocí funkce fetch_assoc() vytvořím asociativní pole, a to uložím do nové proměnné. Poté kontroluji, zda se identifikátor z dotazu nerovná identifikátoru cesty, který přišel z formuláře. Pokud není roven, pokračuji znovu s novým identifikátorem z pole. Pokud už nezbývá v poli žádný nový identifikátor, vypíše se uživateli hláška, která ho informuje o tom, že nelze přidat místo trasy do trasy patřící jiné společnosti. Pokud se identifikátory rovnají, provede se přidání hodnot z formuláře do tabulky Route_Points a uživateli se vypíše hláška o úspěšném přidání.

Třetí výjimka nastává při mazání místa trasy. Pro smazání místa trasy používám dva dotazy. Jeden z nich vybere z tabulky Route_Points identifikátor trasy, kde se identifikátor místa trasy rovná identifikátoru přijatého z formuláře. Druhý dotaz vybere všechny identifikátory tras z tabulky Route, kde se identifikátor společnosti rovná identifikátoru, který je uložený v $_SESSION uživatele. Poté v cyklu porovnávám, zda se rovná identifikátor z prvního dotazu s identifikátorem daného místa trasy přijatým z formuláře a výsledek z druhého dotazu uložený do proměnné pomocí funkce fetch_assoc() ukládám jako asociativní pole do nové proměnné. V cyklu následně ověřuji, jestli se identifikátor trasy, který je v poli, nerovná identifikátoru trasy přijatým z formuláře. Pokud se nerovná, zkouší se další identifikátor z pole. Pokud se ani jeden identifikátor nebude rovnat danému identifikátoru trasy, uživateli se vypíše hláška, která ho informuje, že nejde smazat místo trasy jiné společnosti. Pokud se ale identifikátory rovnat budou, provede se dotaz pro vymazání místa trasy z tabulky Route_Points, kde identifikátor místa trasy je roven identifikátoru místa trasy zaslaným z formuláře, a uživateli se zobrazí hláška o úspěšném vymazání.

Čtvrtá a poslední výjimka nastává v případě úpravy trasy, kdy ověřování probíhá stejně jako u třetí výjimky. V případě úspěchu se uživateli zobrazí informace o úspěšné úpravě místa trasy. V opačném případě se uživateli zobrazí informace, že nejde upravovat místa, která nepatří do jeho společnosti.

3.3.6 Přidání místa do databáze

Jak jsem již zmínil v sekci 3.2.4, je přidání místa do databáze realizováno v souboru insert_dest_db.php. V tomto souboru ověřuji data, která přišla z formuláře ze stránky pro přidání místa. Kontroluji, zda název místa není prázdný a zeměpisná šířka a zeměpisná délka jsou čísla a hodnoty nejsou prázdné. V případě, že by alespoň jedna z podmínek nebyla splněna, uživatel by byl požádán o doplnění informací. Pokud je vše v pořádku, kontroluji, zda uživatel je správce, viz sekce 3.3.4. Pokud je ověření splněno, pomocí dotazu do databáze přidávám zeměpisnou šířku, zeměpisnou délku, název místa, identifikátor společnosti uživatele, který je uložený v $_SESSION uživatele a informaci o místu. Pokud přidání proběhlo v pořádku, je uživateli oznámeno, že přidání místa proběhlo úspěšně.

3.3.7 Úprava a mazání místa v databázi

Úprava místa je realizována v souboru update_dest_db.php. Po odeslání požadavku pro úpravu se kontrolují jednotlivá data, která byla ve formuláři odeslána, a následně se zapisují do proměnných. Poté probíhá kontrola, zda nejsou proměnné prázdné, a zda je uživatel správce, viz sekce 3.3.4. Pokud by nějaká informace chyběla, vyskočí uživateli hláška se žádostí o doplnění všech informací. Toto ověření probíhá na serverové straně a je zde proto, kdyby se správce společnosti pokusil upravit zdrojový kód stránky, protože ověření se nachází i na stránce pomocí parametru required, který je součástí elementu input. Pokud kontrola projde, přejde se ke kontrole společnosti a následné úpravě, viz sekce 3.3.5. Smazání místa je provedeno v souboru del_place_db.php, kdy se kontroluje, zda identifikátor, který z formuláře přišel, není prázdný, a zda uživatel je správce, viz sekce 3.3.4. Poté se pokračuje ověřením společnosti a následným smazáním, viz sekce 3.3.5.

3.3.8 Správa trasy v databázi

V sekci 3.2.6 jsem se zmínil o souborech, které pracují s databází v rámci správy trasy. Z tohoto důvodu zde podrobněji popíši, jejich funkci.

Vymazání trasy se provádí v souboru del_route_db.php. V tomto souboru probíhá po přijetí idenentifikátoru z formuláře pro smazání kontrola, zda není prázdný. Pokud by prázdný byl, bude správce odkázán zpět na stránku pro přidání trasy. V opačném případě proběhne kontrola správce, viz 3.3.4, a pokud je kontrola splněna, přejde se na kontrolu společnosti a následné smazání trasy a jejich míst, viz sekce 3.3.5.

O přidání trasy se stará soubor insert_route_db.php, který zjišťuje, zda hodnota názvu, který přišel z formuláře, není prázdná. Pokud by prázdná byla, byl by uživatel odkázán na stránku pro přidání trasy. Pokud je vše v pořádku, proběhne kontrola správce, viz sekce 3.3.4 a následně se provede dotaz pro přidání trasy do databáze, který do tabulky Route přidává identifikátor společnosti, který je uložený v $_SESSION uživatele, a název cesty. V případě, že vše proběhlo úspěšně, uživateli se vypíše hláška o úspěšném přidání.

Úprava názvu trasy se provádí v souboru update_route_db.php, kdy se kontrolují data, která přišla z formuláře. Z formuláře se odesílá identifikátor trasy a její název. Proto je nutné ověřit, zda žádná z hodnot není prázdná, jinak je uživatel vyzván o doplnění informací. Následně probíhá kontrola uživatele, viz sekce 3.3.4. V případě úspěsné kontroly je prováděna kontrola společnosti, viz sekce 3.3.5. Pokud vše proběhlo v pořádku, uživatel je přesměrován zpět na stránku pro přidání trasy a je informován o úspěšné úpravě.

3.3.9 Správa místa trasy v databázi

V této sekci je podrobný popis souborů, o kterých jsem se zmínil v sekci 3.2.7, a které se starají o správu místa trasy v databázi.

Přidání místa trasy do databáze se provádí v souboru insert_route_point_db.php. V tomto souboru probíhá kontrola hodnot odeslaných z formuláře pro přidání trasy. Kontroluje se, zda hodnoty nejsou prázdné a zda jsou identifikátor, zeměpisná šířka a délka číselné hodnoty. Pokud kontrola proběhla úspěšně, následuje kontrola správce, viz 3.3.4 a následně kontrola společnosti a přidání místa trasy do databáze, viz sekce 3.3.5.

Úprava informací místa trasy v databázi se provádí v souboru update_route_point_db.php. Zde probíhá ověření odeslaných dat formuláře, kdy se kontroluje, zda hodnoty nejsou prázdné a zda identifikátor, zeměpisná šířka a zeměpisná délka jsou číselné hodnoty. Dále probíhá ověření správce, viz sekce 3.3.4, a po úspěšném ověření kontrola společnosti a úprava informací o místě trasy, viz sekce 3.3.5.

Smazání místa trasy je realizováno v souboru insert_route_point_db.php. Zde se ověřuje, zda přijaté data z formuláře nejsou prázdné a zda jednotlivé identifikátory jsou číslo. Následně probíhá kontrola správce, viz sekce 3.3.4, a po správném ověření kontrola společnosti a následné smazání místa trasy, viz sekce 3.3.5.

3.3.10 Správa uživatelů v databázi

Protože je aplikace stavěna na principu uživatelů, je nutné tyto uživatele také spravovat. Správce má možnost aktivovat uživatele, kteří se registrovali do jeho společnosti, a popřípadě deaktivovat uživatele, kteří ze společnosti odešli a neměli by tedy mít nadále do aplikace přístup. O aktivaci se stará soubor update_users_db.php, ve kterém se kontrolují data dodaná ze skrytého formuláře na stránce pro správu uživatelů. Z formuláře dostávám identifikátor uživatele a stav, který se má nastavit v databázi. Protože hodnota 1 v sloupci Approved tabulky users znamená aktivovaného uživatele, kontroluji, zda hodnota, která byla odeslána z formuláře, odpovídá hodnotě 1 a zároveň, zda není hodnota prázdná nebo jiného formátu než je číslo. Číselnou kontrolu a kontrolu prázdné hodnoty provádím i u identifikátoru. Pokud by některá z hodnot byla prázdná, popřípadě by hodnota Approved odeslaná z formuláře byla jiná než 1, vypíši uživateli chybovou hlášku o špatném vstupu. Pokud je vše v pořádku, přejde se ke kontrole správce, viz sekce 3.3.4 a následně kontrole společnosti, viz sekce 3.3.5. Pokud kontrola proběhne v pořádku, provede se dotaz, který nastaví hodnotu Approved z tabulky users na 1, kde identifikátor uživatele je roven identifikátoru z formuláře. Následně se již aktivovaný uživatel může přihlásit do aplikace. O deaktivaci uživatele se stará soubor update_users_db.php, ve kterém je stejný postup jako u aktivace, s výjimkou, že kontroluji, zda hodnota došlá z formuláře, je různá od nuly a je zároveň číslo. Dále postup provádím pro hodnotu 0, která označuje neaktivního uživatele.

3.3.11 Výpis bodů trasy z databáze pro práci s mapou

Na stránce místa společnosti využívám funkce, které pracují s jednotlivými body trasy z databáze, a pro práci s nimi potřebují jejich seznam. O vytvoření seznamu, který obsahuje body jednotlivých tras společnosti se stará soubor xml_route_markers.php. Tento soubor vytváří

XML, ve kterém je seznam jednotlivých míst pro danou společnost. V souboru kontroluji, jestli je uživatel správce. Pokud ano, pomocí dotazu vyberu z databáze z tabulky RoutePoints jednotlivé informace, které potřebuji pro zobrazení bodu a informací o něm. Tabulku RoutePoints spojuji s tabulkou Route pomocí identifikátorů, abych mohl spojit body s danou cestou a její společností. Využívám tedy podmínku pro vypsání bodů, kde identifikátor společnosti je roven identifikátoru společnosti, který je uložený v $_SESSION uživatele. Tento výsledek dotazu si uložím do proměnné. Dále kontroluji pomocí podmínky, zda má výsledek alespoň jeden řádek. Pokud ano, v cyklu pomocí funkce fetch_assoc() uložím jednotlivé řádky do proměnné a vypisuji dané hodnoty z dotazu pro každý řádek. Těmto řádkům s vypsanými hodnotami je přiřazena značka marker a všem řádkům je nadřazená značka markers, ve které jsou jednotlivé značky uzavřeny. Tento výsledný seznam bodů poté využívám ve funkci pro zobrazení trasy, viz sekce 3.4.10.

3.3.12 Výpis míst společnosti z databáze pro práci s mapou

Pro zobrazení jednotlivých míst společnosti na mapě potřebuji jejich seznam, o kterém jsem se zmínil již v sekci 3.3.11. O vytvoření seznamu se stará soubor xml_markers.php. Princip je stejný jako u seznamu pro body tras, až na dotaz, který vybírá z databáze z tabulky Mark všechny informace o místu, kde identifikátor společnosti je roven identifikátoru společnosti, který je uložený v $_SESSION uživatele. Takto vytvořený seznam následně používám pro automatické doplňování míst společnosti na stránce místa společnosti, viz sekce 3.4.8, a pro zobrazení míst společnosti na mapě, viz sekce 3.4.9.

3.3.13 Zapomenuté heslo a jeho obnova

Protože se často stává, že uživatel zapomene heslo ke svému účtu, rozhodl jsem se implementovat funkci, kde si uživatel bude moci vyžádat obnovení svého hesla, na základě e-mailu, který uvedl při registraci. Pokud uživatel zapomene heslo, má možnost využít formuláře na stránce forgotten_password.php, kde zadá svůj e-mail a stiskne tlačítko odeslat žádost.

Po odeslání informací z formuláře se o zpracování žádosti stará soubor send_reset.php. V tomto souboru kontroluji zda e-mail, který došel, není prázdný a poté z něj odstraňuji všechny nedovolené znaky a kontroluji, zda je e-mail platný. Následně v dotazu vyberu všechny údaje o uživateli při splnění podmínky, že e-mail, který došel z formuláře, odpovídá e-mailu uživatele z databáze. Pokud by dotaz nevrátil žádný výsledek, znamenalo by to, že žádný takový e-mail, který došel z formuláře, není v databázi. V tom případě by se uživateli vypsala hláška, která by uživateli oznámila, že zadaný e-mail neexistuje. Pokud ale dotaz nějaký výsledek vrátí, pokračuji dále a pomocí funkce mysqli_fetch_assoc() si do proměnné ukládám informace o uživateli z výsledku dotazu jako pole. Následně si vytvářím proměnnou, do které pomocí PHP funkce mktime() zapisuji aktuální čas odeslání žádosti o změnu hesla plus jeden den. Tento jeden den přidávám, aby uživatel měl po odeslání žádosti o změnu hesla den na to, aby změnu provedl.

Poté si pomocí PHP hashovací funkce md5() vytvářím klíč, který je učený pro daný e-mail a ukládám ho do proměnné. Následně, aby klíč byl unikátní, vytvářím novou proměnnou a pomocí funkce substr() vyberu řetězec dlouhý 10 znaků z funkce md5(), do které vnořuji funkci uniqid(). Tato funkce slouží k vytvoření unikátního identifikátoru na základě času v mikrosekundách, které jako parametr zadávám funkci rand(), která generuje náhodné číslo a hodnotu 1, která určuje, že se vrátí řetězec o 23 znacích. Funkci rand() přidávám z důvodu odlišení kliců, pokud by ve stejnou mikrosekundu vygenerovala funkce 2 klíče najednou. Poté takto vytvořený klíč přidávám k původnímu klíči. Následně pomocí dotazu vkládám do tabulky pass_reset, viz sekce 3.1.3, identifikátor uživatele, vytvořený klíč a datum expirace žádosti o obnovení.

Po vložení informací do databáze ukládám do proměnné naformátovaný text, který je odeslán uživateli. Do tohoto textu přidávám odkaz směřující na soubor reset_password.php, kterému do parametru odkazu přidávám vytvořený klíč a e-mail z formuláře. Pro odeslání e-mailu používám knihovnu PHPMailer, která patří mezi světově nejoblíbenější pro odesílání emailů v PHP. Na hostitelském serveru Endora.cz jsem vytvořil pro doménu e-mailový účet, který knihovna PHPMailer pro odeslání e-mailu požaduje. Pomocí konstruktoru PHPMailer() vytvářím objekt a zajišťuji odeslání e-mailu na e-mail uživatele protokolem SMTP. Základními parametry, které vytvořenému objektu určuji, jsou hostitelský server, ověření pomocí uživatelského jména a hesla pro e-mail na serveru hostitele, zabezpečení komunikace pomocí SSL, port, adresu a název odesílatele, předmět e-mailu, tělo e-mailu a komu se e-mail posílá. Následně se e-mail odešle příjemci.

Po rozkliknutí odkazu v e-mailu, je uživatel přesměrován na stránku reset_password.php. Po otevření pomocí PHP kontroluji, zda z odkazu e-mailu dorazily všechny parametry a přiřazuji je do proměnných. Následně pomocí dotazu vybírám identifikátor uživatele, klíč, e-mail a expirační dobu klíče z tabulky pass_reset, kterou spojuji s tabulkou users na základě identifikátoru uživatele, kde se kontroluje, zda klíč a e-mail, který přišel z odkazu je roven klíči a e-mailu v databázi. Poté kontroluji, zda dotaz vrátil nějaký výsledek nebo ne. Pokud ne, vypíšu uživateli informaci, že odkaz je neplatný, expiroval, nebo již změna hesla byla provedena. Pokud dotaz vrátil výsledek, uložím vybrané informace do proměnné a v podmínce kontroluji, zda je expirační doba stále větší, než je aktuální datum. Pokud ano, uživateli zobrazím formulář pro změnu hesla. V tomto formuláři musí uživatel zadat 2x stejné heslo. Pomocí JavaScriptu kontroluji, zda zadaná hesla odpovídají. Pokud ano, měním CSS styl pole na zelené ohraničení a povoluji uživateli stisknutí na tlačítko pro odeslání změny. Pokud hesla neodpovídají je pole pro zadávání hesla označené červeně a na tlačítko nejde kliknout. Pokud by ale aktuální datum bylo větší než datum expirace, uživateli by se vypsala hláška, která by ho informovala o tom, že odkaz pro obnovu expiroval.

O změnu hesla se stará soubor reset_pass_db.php. V tomto souboru ukládám došlá data z formuláře a kontroluji, zda jsou hesla stejná. Následně pomocí funkce password_hash() heslo zašifruji, tak jako při registraci uživatele, viz sekce 3.3.2, a pomocí dotazu provádím změnu v tabulce users, ve které nastavím heslo uživateli na nové heslo, kde e-mail uživatele a identifikátor uživatele jsou shodné s daty, která přišla z formuláře pro změnu hesla. Pokud je vše v pořádku,

provádím ještě pomocí dotazu smazání všech řádků z tabulky pass_reset, kde se identifikátor uživatele rovná identifikátoru z formuláře. Nakonec pak uživateli zobrazím informaci o úspěšné změně hesla.

3.4 Práce s mapou

V této sekci podrobněji rozeberu funkce pro práci s mapou, které používám a zobecním základní principy pro práci s Google Maps API. V aplikaci používám 2 různé soubory pro práci s mapou na stránkách Mapy a Místa společnosti, a proto tato sekce bude rozdělena do 2 částí, kdy v první části popíši práci s mapou na stránce Mapy a v druhé části práci s mapou na stránce Místa společnosti. Pro stránku Mapy využívám soubor Maps1.js a pro stránku Místa společnosti používám soubor Maps2.js.

3.4.1 Načtení mapy

O načtení mapy se na stránkách stará funkce initMap(), na kterou se odkazuji v rámci stránky Mapy v souboru nearbysearch.php, viz sekce 3.2.9, a v rámci stránky Místa společnosti v souboru corpsearch.php, viz sekce 3.2.10. Tato funkce potřebuje pro zobrazení mapy definovanou pozici, na které se mapa zobrazí. Dále také obsahuje parametr center, kterým určuji, že se mapa zobrazí na střed dané polohy a nakonec je zde parametr zoom, který určuje přiblízení mapy. Základní poloha, při které se mapa načítá, je nastavena na zeměpisnou šířku a zeměpisnou délku uživatele, která je uložená v $_SESSION uživatele. Poté je uživatel vyzván o povolení automatického určení polohy. V případě, že uživatel akceptuje, přesune se mapa na aktuální polohu uživatele.

3.4.2 Určení polohy

Pro určení polohy uživatele používám W3C API pro geolokaci, pomocí které žádosti navigator.geolocation.getCurrentPosition získávám od uživatele po jeho povolení použít aktuální polohu [27]. Pokud uživatel určení polohy povolí, do proměnných si ukládám zeměpisnou šířku a délku. Následně pomocí konstruktoru třídy google.maps.LatLng, který vyžaduje jako parametry zeměpisnou šířku a délku, vytvářím objekt, který následně použiji jako pozici pro vytvoření značky na mapě, viz 3.4.3. Tento stejný postup používám pro tlačítko určení polohy na obou stránkách s mapou, kde pomocí metody addEventListener přiřadím tlačítku reakci na stisknutí, která zavolá funkci, jež provede výše zmíněné určení polohy uživatele a vytvoří značku pro jeho aktuální polohu.

3.4.3 Vytvoření značky

Pro vytvoření značky na mapě používám konstruktor třídy google.maps.Marker, který požaduje jeden parametr typu marker options. Tento parametr specifikuje výchozí nastavení této třídy. Jako požadovaný parametr, který je důležitý při vytváření značky je position, který vyžaduje zeměpisnou šířku a délku. Dále je zde nepovinný parametr map, který určuje, na které mapě se má

značka objevit. Pokud by tento parametr nebyl specifikován, značka se vytvoří, ale nebude přiřazena žádné mapě. Později je možné takto vytvořenou značku do mapy zobrazit pomocí metody setMap(). Jednotlivé značky si po vytvoření přidávám do pole. Pro vymazání značky z mapy používám metodu setMap(), kdy jako parametr zadávám hodnotu null. Tuto metodu pro vymazání používám ve funkci, která prochází pole přidaných značek v cyklu, a tím pádem všechny značky z mapy vymaže. Funkce pro vymazání se volá po stisknutí tlačítka smazat body. Jsou zde ještě další parametry jako například animation, který dodá zobrazení značky animaci, například spadnutí na mapu nebo také parametr clickable, který určuje, zda je možné na značku kliknout či nikoliv. Pro vytváření značek například pro zobrazení míst jsem vytvořil funkci createMarker, abych ji později mohl pro vytvoření více značek na mapě využívat. Funkce obsahuje vytvoření značky pomocí konstruktoru, přidání reakce na stisknutí pomocí metody AddEventListener, kde se provede přiřazení informačního okna, které obsahuje název místa, adresu, webovou stránku, hodnocení a telefon. Tyto informace získávám pomocí metody getDetails(), která zjistí podrobnější informace o místu. Tato metoda je součástí třídy google.maps.places.PlacesService. Tyto získané informace si ukládám do proměnných. Pokud jsou místa, která některé informace neobsahují, provádím kontrolu, zda hodnoty proměnných nejsou definované. Pokud ano, proměnné přiřadím text, který následně v informačním oknu uživateli oznámí, že informace není dostupná.

3.4.4 Vytvoření okna informací

Protože pro některé značky potřebuji vypsat dodatečné informace, které se zobrazí po kliknutí na jednotlivou značku, využívám konstruktor InfoWindow, který využívá objekt InfoWindowOptions určující nastavení pro zobrazení informačního okna. Parametry objektu InfoWindowOptions, které využívám, jsou content a position. Parametr content obsahuje řetězec textu, který se dá také určit pomocí metody setContent() zavolané na objekt InfoWindow. Parametr position využívá zeměpisnou šířku a délku pro určení polohy informačního okna. Protože může být vyskakovací okno přiřazeno objektu typu Marker, bude parametr position nastaven na pozici dané značky. Pro zobrazení daného okna je nutné zavolat na objekt InfoWindow metodu open(), která zobrazí dané informační okno na mapě.

3.4.5 Zobrazení míst v okolí

Pro zobrazení objektů v okolí uživatele využívám knihovnu Places, která musí být definovaná na stránce s mapou. Po zadání okolí v metrech a vybrání typu míst se po stisknutí tlačítka hledat zobrazí daná místa. Toto tlačítko má pomocí metody addEventListener nastavenou reakci na stisknutí, což provede danou funkci. Funkce pro zobrazení míst obsahuje konstruktor třídy google.maps.places.PlacesService, který jako parametr používá mapu pro vykreslování. Dále využívám metodu nearbySearch, která umožňuje hledat místa v určeném rozsahu. Využívá 2 parametry, kterými jsou funkce pro vrácení výsledků a nastavení pro hledání. Jako povinný parametr je lokace, kterou určuji formou kruhové oblasti. Pro toto určení používám parametry

location a radius. Do parametru location zadávám objekt vytvořený z pozice uživatele a do parametru radius zapisuji hodnotu input elementu, do kterého uživatel zadal hodnotu v metrech. Jako poslední využívám parametr type, který určuje typ míst, která se mají zobrazit. Tuto hodnotu získávám z option elementu, kde si uživatel vybral typ míst, která chce zobrazit. Funkce pro vrácení výsledků má 3 parametry, které jsou pole obsahující informace o místech, status, který je vrácený po dokončení vyhledávání výsledků a nakonec objekt, který se používá k zobrazení dalších stran výsledků, pokud by jich bylo více. Již zmíněný status využívám pro ověření podmínky, zda nalezení výsledků proběhlo v pořádku, a pokud ano, v cyklu vytvářím značky na mapě pomocí funkce createMarker(), viz 3.4.3, které dávám jako parametr pole míst a ověřuji, zda existuje více stran výsledků, abych uživateli vypsal všechna nalezená místa.

3.4.6 Navigace

Pro vypočítávání trasy navigace, která může být určena pomocí různých typů dopravy, používám objekt třídy DirectionsService. Tento objekt komunikuje s Google Maps API Directions Service, který po obdržení žádosti o vypočítání cesty vrátí příslušnou cestu. Pro využívání této funkce je nutné využití Directions API, které musí být zapnuté v Google Cloud rozhraní pro daný projekt. V kódu pro možnosti využívat službu vytvářím pomocí konstruktoru objekt třídy DirectionsService. Následně pro vykreslení dané cesty využívám objekt třídy DirectionsRenderer, který vytvářím také pomocí konstruktoru. Do tohoto objektu zadávám nastavení, které bude aplikováno pro vytvořenou cestu. Používám parametr draggable, který zajišťuje, aby se daná cesta nedala po přetažení na mapě změnit. Dále definuji parametr map, který určuje mapu, do které se bude vykreslovat a nakonec parametr panel, který vypisuje informace o cestě do zadaného elementu na stránce. Pro toto zobrazení informací využívám vyjíždějící panel, který se zobrazí po stisknutí tlačítka pro navigaci. Panel se dá rozbalit, popřípadě schovat nebo úplně uzavřít. Pro otevírání a zavírání panelu používám jQuery funkci .slideToggle(), která se spustí po kliknutí na panel. Po vytvoření jednotlivých objektů pro zobrazení mapy používám tlačítko navigovat, kterému jsem přiřadil akci na základě kliknutí. Po kliknutí na tlačítko se volá funkce navigate(), která nastaví navigační panel s informacemi na viditelný. Dále si do proměnných načítám souřadnice na mapě, od kterých se má trasa vytvořit a souřadnice cílové destinace. Jako poslední parametr, který si ukládám, je hodnota z výběru cesty, kterou uživatel pro svou navigaci zvolil. Tyto proměnné následně použiji pro vytvoření žádosti pro vytvoření trasy, která se skládá z atributů origin, destination a travelMode. Tyto atributy značí počáteční bod, koncový bod a způsob dopravy. Nakonec pomocí volání metody directionsService.route(), do které předávám vytvořenou žádost s nastavením a funkci, která obsahuje výsledek metody a její status po provedení. V této funkci pomocí podmínky kontroluji, zda vše proběhlo v pořádku a status je ve stavu OK. Pokud ano, na vytvořený objekt třídy DirectionsRenderer volám metodu setDirections, které předávám parametr výsledku metody directionsService.route a nakonec na objekt volám metodu setMap(), která zobrazí trasu na příslušnou mapu.

3.4.7 Nápověda pro vyhledávání místa

Pro doplňování míst při vyhledávání uživatele využívám knihovnu Places, o které jsem se zmínil již v sekci 3.4.5. Tato knihovna obsahuje třídu SearchBox, kterou využívám pro zobrazování dané nápovědy v textovém poli. Při vytváření objektu této třídy konstruktorem zadávám parametr input, ve kterém je uložený identifikátor daného elementu pro vyhledávání. Dále nastavuji pomocí metody setBounds() preferovanou oblast, do které se mají vracet výsledky nalezených míst. Dále používám funkci, která se zavolá při výběru místa uživatelem z vyhledávacího pole, pokud pro ni existují informace. Pokud byla nějaká místa při zadání uživatele nalezena pomocí metody getPlaces(), uložím je do proměnné jako pole a následně toto pole procházím a využívám pro každé místo funkci createMarker() pro vytvoření značky místa, o které jsem se zmínil v sekci 3.4.3. Nakonec vytvořenou značku zobrazuji na mapu, nastavuji přiblížení mapy a nastavuji mapě hranice pro zobrazení.

3.4.8 Automatické doplňování míst z databáze

Při vyhledávání místa společnosti uživatelem, po zadání názvu místa, nabízím návrhy míst, která jsou aktuálně uložená v databázi společnosti. Tuto nabídku vytvářím za pomoci jQuery metody keyup(), která kontroluje stisky kláves uživatele z HTML input elementu a za pomoci jQuery autocomplete, kde ukládám pole míst ze seznamu a následně seznam uživateli vypisuji. Při stisku klávesy ukládám do proměnné hodnotu z vyhledávacího pole a pomocí objektu XMLHttpRequest načítám soubor se seznamem míst společnosti xml_markers.php, který je popsán v sekci 3.3.12. Následně ve funkci, do které jako parametry zasílám výsledek seznamu z XML souboru a uloženou hodnotu z vyhledávacího pole, procházím řádky pole míst ze seznamu na základě značky a hodnoty ukládám do proměnných. Po uložení všech informací ze seznamu vytvořím proměnnou, do které ukládám název místa ze seznamu a pomocí JavaScript metody substring() odstraňuji z názvu místa všechny znaky, které neodpovídají délce slova z vyhledávání. V podmínce dále ověřuji, zda se část slova z vyhledávání rovná části slova ze seznamu a pokud ano, ukládám informace o místu do proměnných, které následně používám pro zobrazení místa na mapě, viz sekce 3.4.9.

3.4.9 Zobrazení míst z databáze na mapě

Po stisknutí tlačítka hledat při vyhledávání místa společnosti do proměnných ukládám informace o místu, které jsem získal ze seznamu míst společnosti při vybráním místa uživatelem při automatickém doplňování, o kterém jsem psal v sekci 3.4.8. Následně danému místu vytvářím značku, viz sekce 3.4.3, přiřazuji ji do pole značek a vytvářím pro místo informační okno, viz sekce 3.4.4, které se zobrazí po kliknutí na značku. Pohled mapy a její přiblížení nakonec přesunu na danou značku pomocí metod setCenter() a setZoom().

3.4.10 Zobrazení trasy z databáze na mapě

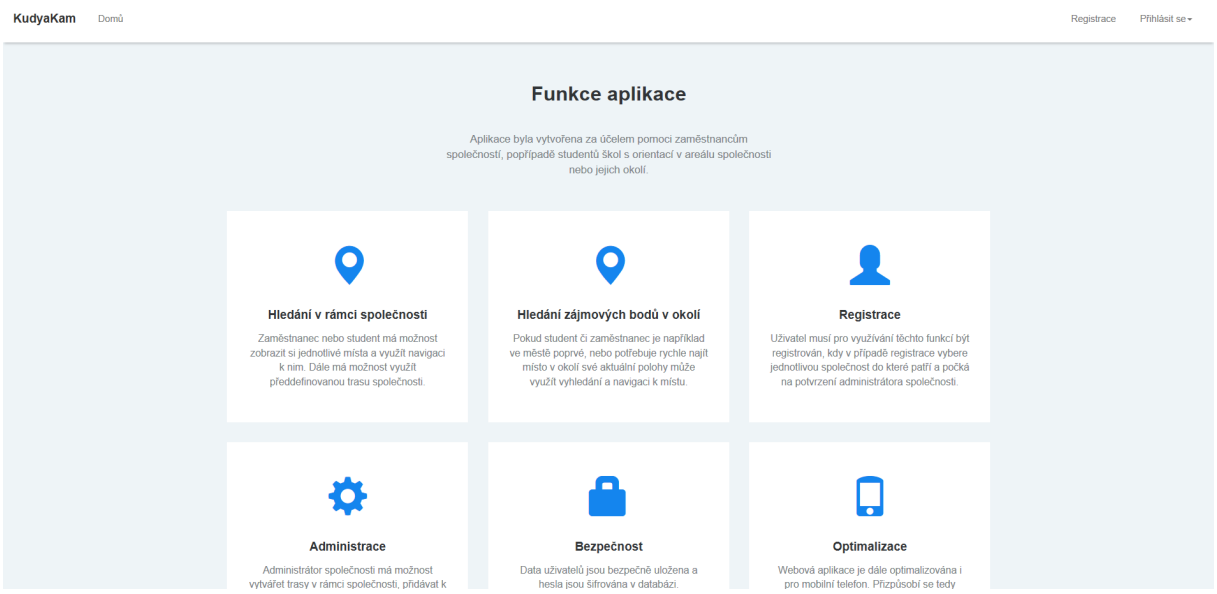
Po stisknutí tlačítka zobraz trasu se zavolá funkce, která využívá soubor xml_route_markers.php, který jsem podrobněji popsal v sekci 3.3.11. Pro načtení XML souboru využívám objekt XMLHttpRequest, který se používá pro výměnu dat se serverem. Tento objekt je základem AJAX programování [26]. Po vrácení hodnot, které byly načteny ze souboru, je pomocí značky marker uložím do proměnné jako pole. Toto pole následně procházím a kontroluji číslo trasy u míst ze seznamu, které odpovídá číslu trasy, vybrané uživatelem na stránce místa společnosti. Po nalezení odpovídajícího čísla trasy u místa do proměnných ukládám informace o daném místu. Po uložení informací vytvářím značku místa a jeho informační okno na základě uložených informací. Průběh vytvoření značky místa je popsán v sekci 3.4.3 a informačního okna v sekci 3.4.4. V dalším kroku přidávám do pole zeměpisné souřadnice daného místa a do proměnných ukládám souřadnice pro počáteční a koncový bod trasy, které poté využívám při zobrazení trasy. Dále se zavolá funkce pro zobrazení trasy na mapě, ve které nastavuji panel pro zobrazení navigace jako viditelný, a do proměnné ukládám způsob dopravy, který si uživatel vybral. Následně ve funkci pomocí metody shift() odeberu z pole míst první bod trasy a pomocí metody pop() poslední bod trasy. Tyto body mám již uložené v proměnných pro počáteční a koncový bod, takže je při výpisu trasy nepotřebuji zobrazit dvakrát. Nakonec ve funkci při vytváření žádosti pro zobrazení trasy a navigaci, viz sekce 3.4.6, zadávám pro začátek trasy souřadnice počátečního bodu a pro konec trasy souřadnice koncového bodu, které jsem si dříve uložil. Poslední parametr, který definuji v žádosti je waypoints, do kterého zadávám pole míst trasy vytvořené ze seznamu. Tento parametr určuje přes jaké body má trasa od začátku do konce procházet. Výsledná trasa s jejími body a informacemi o cestě se následně zobrazí na mapě.

3.4.11 Smazání trasy

Smazání trasy mezi dvěma a více body provádím pomocí tlačítka smazat trasu. Tlačítku je přiřazena funkce, která se provede po jeho stisknutí. Tato funkce nastaví element, ve kterém jsou zobrazeny informace o navigaci na neviditelný. Dále pomocí metody setDirections třídy DirectionsRenderer nastavuji hodnotu routes[] na prázdné pole. Pole route[] je pole obsahující objekty, které nesou informace o jednotlivých cestách. Nakonec pomocí metody setMap(), které zadávám jako parametr hodnotu null, vymažu z mapy zobrazenou cestu, volám funkci pro vymazání bodů z mapy a nastavuji hodnoty pro počáteční bod a koncový bod na 0, aby se při vytvoření nové cesty mohly znovu využít tyto hodnoty.

4 Testování aplikace

Pro dokončení aplikace bylo velmi důležité její testování. Testování bylo provedeno, aby se předešlo, v případě špatného pochopení prvků aplikace nebo špatných vstupů, chybám aplikace nebo také nevyužití jejího plného potenciálu z pohledu uživatele. Oslovil jsem tedy skupinu osob, které byly jak starší tak mladší generace. Někteří z nich byli také zaměřeni na obor informatiky. V rámci testování jsem uživatele požádal o několik úkolů, abych zjistil, jak rychle se v aplikaci dokáží orientovat, využívat vyhledávání v rámci mapy a zvládat její správu z pohledu správce aplikace. Dále jsem po skupině osob s informatickým zaměřením požadoval otestování vstupů aplikace. Uživatelé aplikaci testovali jak na mobilních zařízeních se systémy Android i iOS tak ve webových prohlížečích na stolním počítači. Jako první bylo provedeno testování ve webovém prohlížeči Google Chrome a Microsoft Edge na stolním počítači a po otestování ve webovém prohlížeči bylo testování provedeno na mobilním zařízení v systémech iOS a Android. Uživatelé si na domovské stránce aplikace přečetli, co aplikace umí a obsahuje. To je znázorněno na obrázku 3. Protože uživatelé zjistili, že je nutné být pro využití aplikace registrován, přešli tedy k registraci. Uživatelé neměli problém s registrací a na první pohled pochopili, že je nutné vybrat nějakou společnost, do které se chtějí registrovat. Po registraci se někteří uživatelé snažili přihlásit, ale bylo jim oznámeno, že jejich účet není aktivovaný. Z tohoto důvodu, jsem po úspěšné registraci přidal hlášku, která uživateli oznámí, že musí vyčkat na schválení správcem.

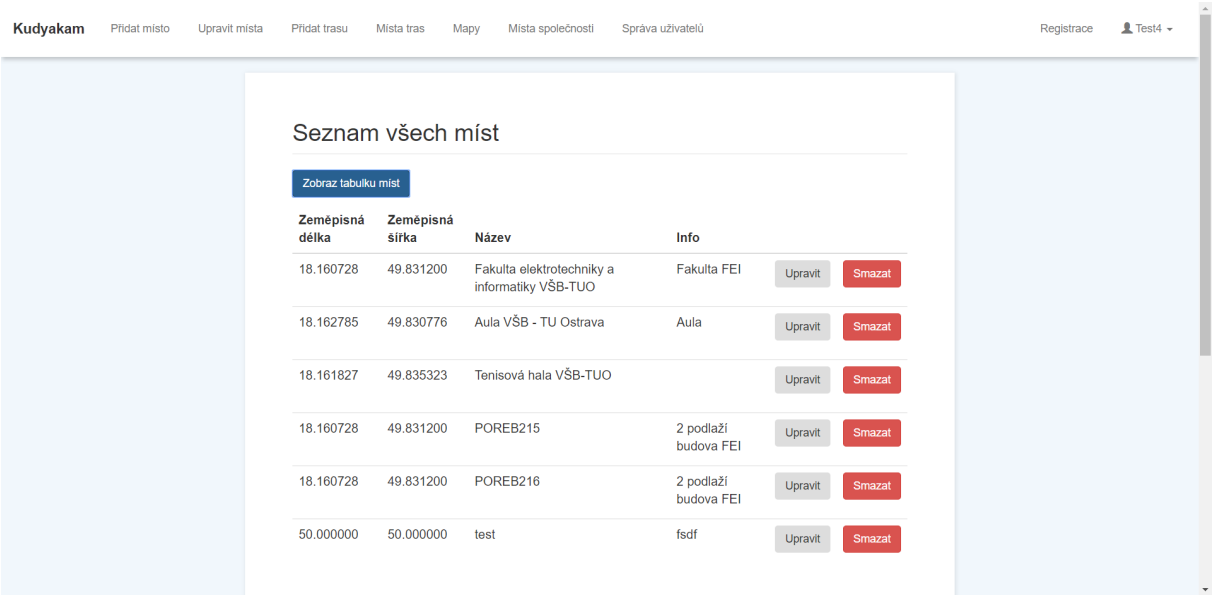


Obrázek 3: Domovská stránka aplikace v prohlížeči Microsoft Edge

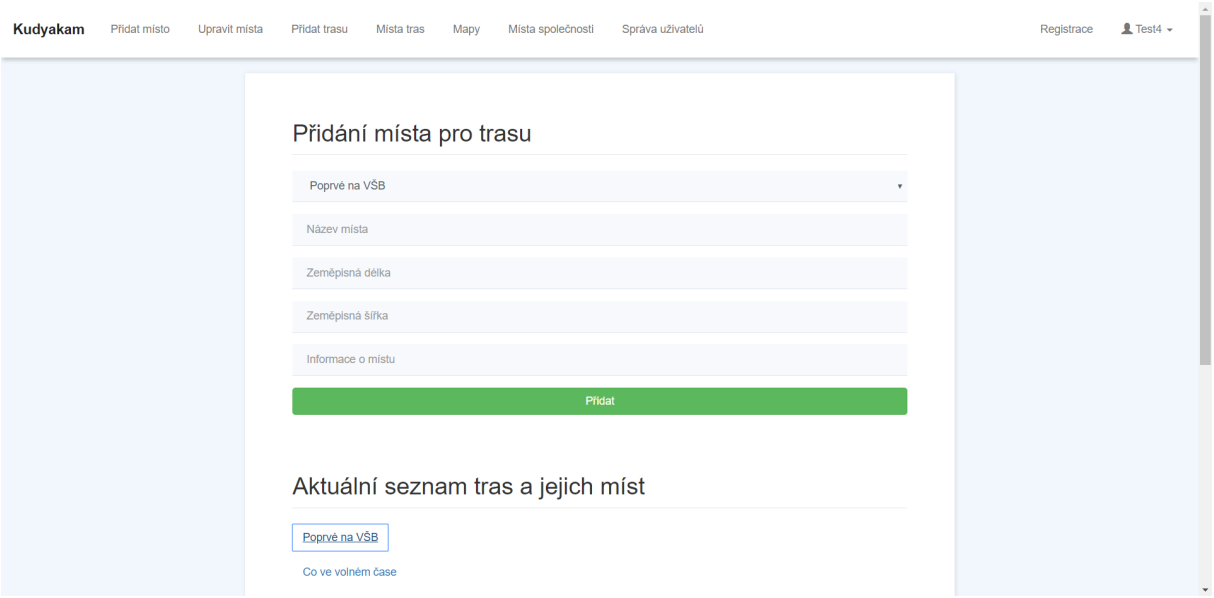
4.0.1 Testování z pohledu správce

Vybral jsem tedy dva správce pro dvě společnosti a ti měli za úkol aktivovat uživatele, kteří se právě registrovali. Správci po přihlášení do aplikace byli schopni ihned aktivovat nové uživatele,

díky přehlednému menu a správným popiskům. Poté jsem správce požádal, aby přidali nějaké místo pro svou společnost. Správci tedy přešli na stránku pro přidání místa. Díky přehledným popiskům jednotlivých polí formuláře byli správci schopni vyplnit formulář pro přidání. Jeden ze správců ale prohodil zeměpisnou šířku a délku místa. Proto jsem ho vyzval, aby se místo pokusil upravit. Přešel tedy na stránku upravit místa a zobrazil si tabulku všech míst, což je vidět na obrázku 4. Pomocí tlačítka upravit si svoje místo byl schopen upravit pomocí formuláře pro aktualizaci místa. Poté jsem vyzval správce o přidání trasy společnosti. Správcům z počátku nebylo jasné, proč se trasa přidává. Vysvětlil jsem jim tedy, že jednotlivým trasám, které si vytvoří, mohou přidat body, které se následně jako trasa zobrazí uživateli na mapě, něco jako například okruh. Po vysvětlení si tedy správci vytvořili trasu a dále jsem je vyzval o přidání míst k jimi vytvořené trase. Přešli tedy na stránku místa tras, která je zobrazena na obrázku 5, a pomocí přehledného formuláře si vybrali svou vytvořenou trasu a vytvořili pro ni místo. Poté jsem je vyzval o vytvoření ještě dalších dvou míst, aby se trasa skládala ze tří bodů. Správci tak provedli a následně jsem je vyzval, aby si zobrazili body trasy, které přidali. Přešli tedy k aktuálnímu seznamu tras a jejich míst a zobrazili si svou trasu s místy. Po dotazu, zda by byli schopni místa upravit či smazat, oba odpověděli ano. Následně jsem je vyzval, aby si ověřili, že jejich trasa a místo, které přidali, se zobrazí na mapě společnosti. Správci tedy přešli na stránku místa společnosti, kde pomocí vyhledávacího okna místo našli a zobrazili na mapě. Dále si správci měli možnost vybrat svou trasu, kterou si vytvořili a zobrazit si ji na mapě. Vyzkoušeli si také navigaci k místu. Po otestování jsem je požádal o zpětnou vazbu k aplikaci. Podle jejich slov neměli problém s administrací aplikace a pochopili, jak by se dále o aplikaci starali. Přehlednost stránky hodnotili kladně a líbil se jim jednoduchý vzhled aplikace. Následně jsem správce požádal o vykonání stejných úkolů na jejich mobilním zařízení. Správci neměli problém se správou míst a celkovou správou aplikace. Pozitivně hodnotili responzivitu webové aplikace na mobilním zařízení. Dále hodnotili pozitivně správu aplikace a přehlednost formulářů na mobilním zařízení. Jediné výtky měli k zobrazení mapy, kde stránka je rozdělena na panel s vyhledáváním, který je nahoře, a pro mapu museli přejít níže v rámci stránky. Zobrazení aplikace na mobilním zařízení, je vidět na obrázcích 7a a 7b.



Obrázek 4: Zobrazení seznamu míst společnosti v prohlížeči Google Chrome

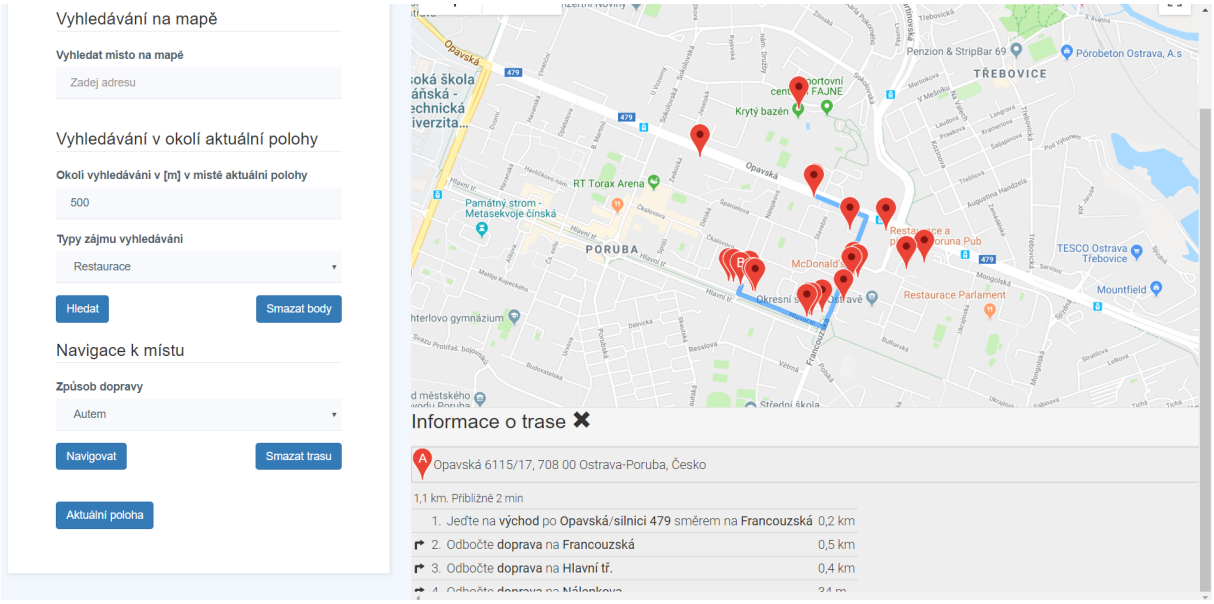


Obrázek 5: Přidání místa trasy společnosti v prohlížeči Google Chrome

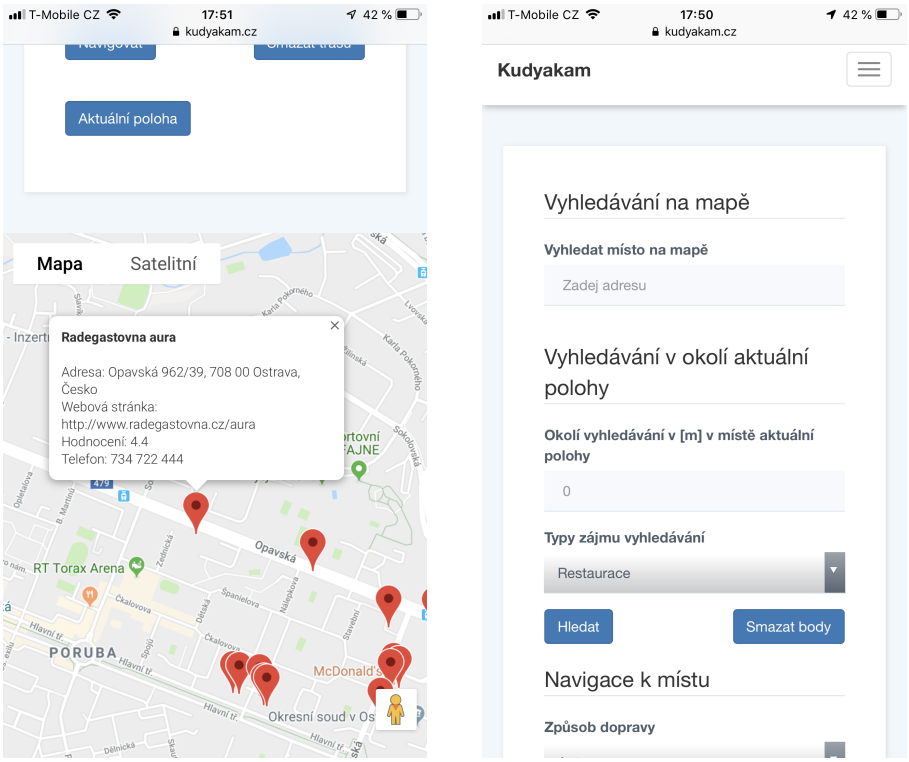
4.0.2 Testování z pohledu uživatele

Následně jsem vyzval uživatele, kteří se registrovali a správci je aktivovali, o přihlášení do aplikace. Tyto uživatele jsem vyzval, aby přešli na stránku mapy a vyzkoušeli najít nějaké místo. Každý zadal nějakou adresu a místo si na mapě zobrazil. Také jsem vyzval uživatele, aby vyzkoušeli navigaci ke svému místu. Všichni si poté úspěšně zobrazili informace o trase v dolním panelu. Poté jsem uživatele vyzval, aby si vyzkoušeli najít restaurace v okolí 500 metrů v rámci

své aktuální polohy. Výsledek vyhledávání je vidět na obrázku 6. Všichni uživatelé pochopili, že do pole pro okolí vyhledávání musí zadat zmíněných 500 metrů a vybrat restaurace. Všem se povedlo zobrazit si dané restaurace a po kliknutí na některé z bodů na mapě si zobrazili informace o restauraci. Po odzkoušení funkcí jsem uživatele vyzval o přesunutí na stránku místa společnosti, kde jsem uživatele požádal o vyhledání míst, které přidali správci pro svou společnost, a vysvětlil jsem jim, že správci přidali místo pro svou společnost a díky tomu ji mohou vyhledat na mapě. Vysvětlil jsem jim, že takto by mohli vyhledávat v rámci společnosti všechna místa, budovy a kanceláře v areálu. Po vysvětlení uživatelé zhodnotili, že by určitě v rámci podniků a společností byla tato funkce užitečná pro orientaci ve společnosti. Následně jsem uživatele vyzval o zobrazení trasy společnosti. Všichni si zobrazili trasu a informace o místech trasy. Zobrazení trasy je vidět na obrázku 1. Nakonec jsem uživatele požádal o zhodnocení aplikace. Všichni uživatelé hodnotili aplikaci kladně. Uživatelům se líbilo zejména rozložení stránky pro mapy, kde využili levého panelu pro funkce aplikace a hned mohli výsledek vidět na mapě. Ocenili dále možnost zobrazení informací o místech po kliknutí na bod mapy a líbila se jim možnost zobrazení trasy společnosti. Uživatele jsem následně vyzval o otestování aplikace na svém mobilním zařízení. Uživatelé vykonali stejné úkoly, jako při testování ve webovém prohlížeči a všechny úkoly splnili. Po otestování jsem je požádal o zpětnou vazbu. Jako u testování u správců mi uživatelé sdělili dobrou přehlednost aplikace na mobilním zařízení. Pozitivně hodnotili menu a přizpůsobení aplikace mobilnímu zařízení. Opět ale měli malé výtky k zobrazení map, které jsou umístěny pod vyhledávacím polem.



Obrázek 6: Vyhledávání v okolí v prohlížeči Google Chrome



(a) Zobrazení informací o místě (b) Vyhledávání na mapě

Obrázek 7: Zobrazení aplikace na mobilním zařízení

4.0.3 Testování vstupů aplikace

Po odzkoušení aplikace klasickými uživateli, jsem vyzval kolegu se zaměřením na informatiku, o otestování vstupů aplikace v rámci formulářů. Požádal jsem jej o registraci, kdy se ihned pokusil odeslat registraci s nevyplněnými údaji. To se mu nepovedlo, protože toto mám ošetřené jak na stránce aplikace tak na serverové straně. Vyzkoušel tedy formulář vyplnit, ale otevřel si zdrojový kód, ve kterém smazal číslo společnosti u prvku při výběru společnosti a formulář odeslal. Aplikace mu vypsala žádost o doplnění všech informací. Zkusil proto zadat náhodné číslo, které opět bylo chybné. Formulář odeslal a vyskočila mu hláška o chybě při vytváření. Oba tyto problémy jsou ošetřeny na serverové straně. Registroval se tedy s platnými údaji do konkrétní společnosti. Následně jsem jej aktivoval a udělil mu přístup správce pro společnost. Poté jsem jej požádal o otestování vstupů pro přidání místa. Vyzkoušel proto do zeměpisné šířky a délky napsat text. To se mu ale nepodařilo, díky kontrole čísla. Vyzval jsem jej tedy o vyzkoušení úpravy místa. Zobrazil si seznam míst a otevřel si zdrojový kód stránky. Zde si našel prvek pro smazání stránky a přepsal hodnotu identifikátoru na hodnotu, která patří jiné společnosti a stiskl tlačítko pro smazání. V tu chvíli se mu podařilo smazat místo jiné společnosti. Následně vyzkoušel to samé pro úpravu. Oba pokusy byly úspěšné. Poté jsem jej požádal, aby to stejné odzkoušel u trasy, místa tras a u správy uživatelů. Všechny tyto následné pokusy byly úspěšné.

Z tohoto důvodu jsem musel na serverové straně ošetřit i tuto záležitost a implementovat ověření společnosti, které popisuji v sekci 3.3.5. Nakonec jsem jej požádal o otestování ostatních vstupů. Ostatní vstupy byly už v pořádku a pečlivě ošetřeny. Za toto otestování mu ovšem velmi děkuji, protože by mě samotného nenapadlo, že by někdo něco podobného mohl zkoušet.

Závěr

V rámci této práce jsem vytvořil responzivní webovou aplikaci a její serverovou část. Během vývoje jsem se snažil zaměřit na to, aby aplikace jako taková, byla uživatelsky přívětivá, a aby byla využitelná především v praxi. Odzkoušel jsem si, jak probíhá návrh takové aplikace a jaké kroky je nutné učinit pro to, aby byla aplikace co nejlépe stavěná v rámci použitých technologií. V rámci práce jsem se naučil pracovat s Google Maps API a propojit mapu s vlastní databází. Pomoci analýzy požadavků na aplikaci jsem se snažil do aplikace zakomponovat všechny prvky, které by uživatel či správce aplikace mohl využít. Vytvořil jsem tedy systém pro správu míst společností, jejich tras a míst, které jsou určeny pro danou trasu. Dále jsem úspěšně navrhl administraci uživatelů společností. Pomocí Google Maps API jsem implementoval do aplikace funkčnost mapy a práci s mapou v rámci společnosti s využitím vlastní databáze a vyhledávání zájmových míst v rámci okolí uživatele aplikace.

V průběhu vývoje jsem narazil na několik problémů, které se mi podařilo úspěšně vyřešit. Při vývoji a testování aplikace jsem zjistil, že je nutné ošetřit všechny vstupy aplikace, aby nedocházelo v aplikaci ke zbytečným chybám a neoprávněným úpravám ze strany uživatele. Další z problémů, který jsem musel řešit, byl prvotně špatný návrh databáze a jejich tabulek, kdy jsem si při vývoji uvědomil, že je nutné databázi změnit. Tato změna mi zabrala poměrně dost času, protože jsem po změně databáze musel změnit i zdrojový kód. Dalším problémem, na který bych rád upozornil, je kontrola funkcí a vlastností v rámci využití frameworků, popřípadě různých stylů a také programovacích jazyků. Některé verze obsahují funkce a prvky, které zase jiné verze nemají a naopak. Proto je dobré dbát na to, co jaká verze obsahuje. V rámci vývoje jsem musel řešit právě zmíněný problém s verzí frameworku, protože jsem chtěl využít prvky, které v nejnovější verzi už nebyly zakomponovány, a proto jsem následně musel styl aplikace předělat na starší verzi frameworku.

Proces vývoje aplikace jako takový je poměrně časově náročná záležitost, u které si člověk musí rozvrhnout, kde začít, aby nedocházelo ke zbytečným změnám v rámci kódu, popřípadě změně technologií a jejich verzí. Je nutné proto před každým vývojem zjistit, jaké funkce by aplikace měla obsahovat a udělat si rozbor technologií, které pro aplikaci budou použity. V dnešní době je také důležité myslet na celkovou responzivitu aplikace, protože stále více lidí používá častěji mobilní zařízení než stolní počítače.

V rámci vylepšení aplikace do budoucna by mohla aplikace obsahovat funkci pro přidání fotky k místu společnosti a mohlo by dojít k lepšímu zpracování vyhledávání na mapě na mobilním zařízení. Dále by se mohlo přidat vyhledávání v okolí určitého místa, které uživatel zadal při vyhledávání. Dalším prvkem, který by aplikace mohla obsahovat, je propojení s Active Directory, například v rámci škol, a mohla by využívat pro přihlášení uživatele stejné údaje, jako v univerzitním systému. Aplikace by ale dle mého názoru ve stavu, ve kterém aktuálně je, určitě našla uplatnění v praxi v rámci společností, či menších podniků.

Literatura

[1] PROCHÁZKA, David. PHP 6: začínáme programovat. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-3899-4.

[2] SUEHRING, Steve. JavaScript: krok za krokem. Brno: Computer Press, 2008. Krok za krokem (Computer Press). ISBN 978-80-251-2241-9.

[3] CASTRO, Elizabeth a Bruce HYSLOP. HTML5 a CSS3: názorný průvodce tvorbou WWW stránek. Brno: Computer Press, 2012. ISBN 978-80-251-3733-8.

[4] GASSTON, Peter. The Book of CSS3, 2nd Edition. San Francisco: No Starch Press,US, 2014. ISBN 978-1-59327-580-8.

[5] SQL injection. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/SQL_injection

[6] ČÁPKA, David. Úvod do CSS frameworku Bootstrap. ITnetwork.cz [online]. [cit. 2019-04-09]. Dostupné z: https://www.itnetwork.cz/html-css/bootstrap/kurz/uvod-do-css-frameworku-bootstrap/

[7] Bootstrap. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/Bootstrap

[8] MICHÁLEK, Martin. K čemu je dobrý Bootstrap a frontend frameworky? [online]. [cit. 2019-04-09]. Dostupné z: https://www.zdrojak.cz/clanky/k-cemu-je-dobry-bootstrap-frontend-frameworky/

[9] Foundation (framework). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://en.wikipedia.org/wiki/Foundation_(framework)

[10] PHP. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/PHP

[11] JavaScript. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript

[12] PhpMyAdmin. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/PhpMyAdmin

[13] InnoDB. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/InnoDB

[14] MyISAM. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-09]. Dostupné z: https://cs.wikipedia.org/wiki/MyISAM

[15] KOTRÁŠ, Jan. Verzování databáze při vývoji aplikací [online]. Brno, 2017 [cit. 2019-04-10]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=159258. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

[16] POTTER, Jason. MySQL Performance: MyISAM vs InnoDB [online]. [cit. 2019-04-10]. Dostupné z: https://www.liquidweb.com/kb/mysql-performance-myisam-vs-innodb/

[17] Zázemí webhostingu. Endora.cz [online]. [cit. 2019-04-10]. Dostupné z: https://www.endora.cz/o-nas/zazemi

[18] Vlastnosti webhostingu. Endora.cz [online]. [cit. 2019-04-10]. Dostupné z: https://www.endora.cz/vlastnosti

[19] Mapy API verze 4.13 – Neil Armstrong. Mapy.cz [online]. [cit. 2019-04-10]. Dostupné z: http://api.mapy.cz/

[20] Mapové podklady. Seznam.cz [online]. [cit. 2019-04-10]. Dostupné z: https://napoveda.seznam.cz/cz/mapy/mapove-podklady/zakladni-mapovy-podklad/

[21] TEMPRANO, Victor. Google Maps API or Leaflet: What's Best for your Project?. Codementor [online]. [cit. 2019-04-10]. Dostupné z: https://www.codementor.io/victorgerardtemprano/google-maps-api-or-leaflet–what-s-best-for-your-project-faaev60vm

[22] Overview. Leaflet [online]. [cit. 2019-04-10]. Dostupné z: https://leafletjs.com/

[23] Leaflet FAQ. GitHub [online]. [cit. 2019-04-10]. Dostupné z: https://github.com/Leaflet/Leaflet/blob/master/FAQ.md

[24] FAQ. Google Maps Platform [online]. [cit. 2019-04-10]. Dostupné z: https://developers.google.com/maps/faq#whatis

[25] RŮŽIČKA, Vojtěch. OWASP Top Ten 2017. VojtechRuzicka [online]. [cit. 2019-04-10]. Dostupné z: https://www.vojtechruzicka.com/owasp-top-ten-2017/

[26] AJAX Introduction. W3schools.com [online]. [cit. 2019-04-10]. Dostupné z: https://www.w3schools.com/php/php_ajax_intro.asp

[27] Geolocation API Specification 2nd Edition. W3.org [online]. [cit. 2019-04-10]. Dostupné z: https://www.w3.org/TR/geolocation-API/